VALIDACIÓN Y COMPARACIÓN CUANTITATIVA DE LAS TÉCNICAS SPEEDED UP ROBUST FEATURES Y CONVOLUTIONAL NEURAL NETWORKS PARA EL RECONOCIMIENTO DE ARMAS DE DISPARO IDÉNTICAS

VERÓNICA MARTÍNEZ HERNÁNDEZ

KELLY TATIANA MORALES GUEVARA

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

INGENIERÍA FÍSICA

PEREIRA, RISARALDA

2019

Validación y Comparación Cuantitativa de las Técnicas Speeded Up Robust Features y Convolutional Neural Networks para el Reconocimiento de Armas de Disparo Idénticas.

Verónica Martínez Hernández

Kelly Tatiana Morales Guevara

Trabajo de investigación formativa de tipo experimental presentado como requisito parcial para optar al título de ingeniero físico

Director

M. Sc. Jimmy Alexander Cortés Osorio

Universidad Tecnológica de Pereira Facultad de Ingenierías

Ingeniería Física

Pereira, Risaralda

2019

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

PREFACIO

1.1. RESUMEN

La identificación de armas idénticas ha sido una gran necesidad desde que éstas fueron construidas, por tanto, a partir del año 1835 se comenzó la búsqueda de métodos para detectar armas de disparo involucradas en delitos, esto ha permitido que se juzguen los crímenes con la mayor eficacia posible; sin embargo, lo anterior se veía afectado por el prolongado proceso de identificación, además de que dependía de la visión de los examinadores balísticos, quienes por medio de un microscopio de comparación analizaban y clasificaban manualmente las vainillas para dar un dictamen. La vinculación del procesamiento de imágenes a esta tarea, desde hace menos de dos décadas, sirvió sin duda para optimizar el tiempo de identificación, el personal requerido y para obtener mejores resultados. Los estudios realizados acerca de este tema en particular han involucrado diversas técnicas de procesamiento, tales como, sistemas balísticos de imágenes, características numéricas, impresión de la imagen en la totalidad del percutor y herramientas de Microsoft Office; estas técnicas han arrojado resultados de hasta 96 % de efectividad a la hora de clasificar armas no idénticas, con lo que se evidencia que son una alternativa viable para quienes trabajan en el área de balística. En este proyecto de grado se implementarán las técnicas Speeded Up Robust Features y Convolutional Neural Networks, para el reconocimiento de armas de disparo idénticas a partir de una base de datos que contiene 250 vainillas pertenecientes a 5 pistolas, con ellas se busca extraer características identificativas de la base de la vainilla, tales como, huella del eyector, fulminante y huella de percusión; siendo éstas diferentes para cada arma, incluso si se quiere comparar armas con las mismas características de clase (marca, modelo y calibre), las anteriores características identificativas se encuentran contenidas en la base de la vainilla y son dejadas cuando el arma es disparada; la huella del eyector representa valiosa información acerca del arma debido a que es dejada por el eyector (dispositivo que proyecta los cartuchos al exterior) en el momento del disparo. El fulminante, al momento de ser percutido, libera un componente altamente sensible con el cual se produce la explosión. Finalmente la huella de percusión es la impresión resultante en el fulminante a partir del choque de éste con la aguja percutora, contiene información de la forma geométrica de la aguja. Por esta razón estas características

brindan información diferenciable entre las armas.

La primer técnica (SURF) será catalogada por medio de los clasificadores K-Nearest Neighbors (KNN) y Support Vector Machines (SVM) para su posterior validación teniendo en cuenta los parámetros de Sensibilidad, Especificidad y Matriz de Confusión (Tasa de verdaderos positivos, verdaderos negativos, falsos positivos, falsos negativos). Con la segunda técnica (CNN) no es necesario usar minería de datos pues ésta implementa un clasificador propio; sin embargo, a ésta también se anexará su respectiva Matriz de Confusión con los valores de Sensibilidad y Especificidad. Las técnicas serán implementadas en el lenguaje de programación de MATLAB 2018 y arrojarán resultados, después de su clasificación, que permitirán concluir si son viables para este problema particular y si aportan mejoras a los procedimientos de identificación de armas de disparo idénticas.

1.2. INTRODUCCIÓN

Desde el comienzo del Conflicto Armado en Colombia, se han evidenciado múltiples casos de violencia, en los cuales el 81 % involucra armas de fuego [1]. Existen técnicas que facilitan a los equipos de balística la identificación de estas armas en escenas de crímenes; sin embargo, y dado que los grupos policiales en la mayoría de los casos son dotados con armas con las mismas características de clase, el reconocimiento de éstas se hace más complejo. En este caso se aplicarán dos técnicas, las cuales surgen a partir del estudio de artículos que tratan este tipo de problema y que han arrojado resultados óptimos, éstas son: SURF (Speed Up Robust Features) y Convolutional Neural Networks. La primera consiste en la extracción de 64 keypoints a partir de una imagen sin importar su rotación, traslación, escala y/o proyección, éstos serán relacionados con las características identificativas que se pretende encontrar en las armas. La segunda, a diferencia de la técnica anterior, hace uso de tres tipos de capas con el fin de extraer características y clasificar; estas capas son llamadas capas convolucionales, capas de reducción o pooling y capas clasificadoras; las últimas hacen que no sea necesario utilizar un clasificador. Esta técnica es invariante a la traslación, rotación y escala. La intención es comprobar la validez que tienen para el reconocimiento de armas idénticas mediante el estudio de marcas características en la base de la vainilla, las cuales son: huella del eyector, excentricidad y sesgo, textura del cráter de percusión y firma digital en el borde del cráter. Para esto se hará uso, para la primer técnica, de la minería de datos a partir de tres clasificadores conocidos como K-Nearest Neighbors (KNN) y Support Vector Machines (SVM).

1.3. JUSTIFICACIÓN

Dentro de los mecanismos utilizados para el reconocimiento de armas de disparo, ha surgido la aplicación de distintas técnicas de detección de características y se han realizado varios estudios en el área; sin embargo, no se tiene claridad en la precisión de muchas de éstas para la identificación de armas de

disparo idénticas. Tomando como base dos técnicas recogidas debido a sus buenos resultados en estudios anteriores [2], se llegó a la siguiente pregunta:

¿Cuál de los métodos propuestos, SURF (Speeded Up Robust Features) y CNN (Convolutional Neural Networks) para el reconocimiento de armas de disparo idénticas, aplicado a una base de 250 vainillas pertenecientes a 5 pistolas es mejor en términos de las pruebas de Validez?

La solución de casos en los que se involucran armas de fuego idénticas resulta compleja para los equipos de balística de la Fiscalía en Colombia. Dicha entidad pagaba a una compañía para la utilización de un software que contribuía a la identificación de armas. A la larga y por motivo de altos costos, la Fiscalía tomó la decisión de construir su propio software llamado SUCOBA (Sistema Único de Comparación Balística). El procesamiento de imágenes ha brindado una solución a problemas en varias áreas de la ciencia, entre las que se incluye la balística forense [3]. La utilización de técnicas de procesamiento de imágenes para el reconocimiento de armas ha sido estudiada en países como China, India, Estados Unidos y Malasia. El presente proyecto busca analizar la validez de dos de las múltiples técnicas (SURF y CNN) para el reconocimiento de armas de disparo idénticas y, de obtener resultados satisfactorios, exponer estas técnicas como una estrategia factible para el mejoramiento del software que viene siendo desarrollado por la Fiscalía (Seccional Pereira). Al recopilar estudios acerca de la identificación de armas, se puede evidenciar que este problema, bajo técnicas de extracción de características, ha sido resuelto eficientemente, ya que se han expuesto resultados de entre el 74 % y 96 % de eficacia [4] [5] [6], ahora resulta apropiado analizar la eficacia en particular de las dos técnicas anteriormente mencionadas en el reconocimiento de armas idénticas, aún cuando éstas han sido utilizadas en el área de la balística forense [7] [5].

1.4. OBJETIVOS

1.4.1. Objetivo General

Realizar pruebas de Validez y comparar las técnicas SURF (Speeded Up Robust Features) y CNN (Convolutional Neural Networks) para el reconocimiento de armas de disparo idénticas aplicado a una base de datos de 250 vainillas pertenecientes a 5 pistolas, usando los clasificadores K-Nearest Neighbors (KNN) y Support Vector Machines (SVM) para la primer técnica y el clasificador propio de la segunda.

1.4.2. Objetivos Específicos

- Recolectar imágenes de 250 vainillas pertenecientes a 5 pistolas con el fin de crear una base de datos sobre la que se realizará la debida extracción de características.

- Realizar el pre-procesamiento digital de las imágenes obtenidas en la base de datos, con el fin de preparar dichas imágenes para su extracción de características.
- Implementar las técnicas SURF (Speeded Up Robust Features) y CNN (Convolutional Neural Networks) con el fin de extraer las características identificativas a las imágenes de la base de las vainillas y categorizarlas mediante los clasificadores K-Nearest Neighbors (KNN) y Support Vector Machines (SVM) para la primer técnica.
- Interpretar y comparar los resultados obtenidos a partir de la clasificación y sus pruebas de Validez para las técnicas propuestas en el reconocimiento de armas de disparo idénticas.

Capítulo 2

ESTADO DEL ARTE

2.1. BALÍSTICA FORENSE E IDENTIFICACIÓN DE ARMAS DE DISPARO

El primer intento exitoso en el descubrimiento del autor de un crimen realizado con arma de fuego se dio en el año 1835 en Londres (Inglaterra), bajo la primitiva investigación de Henry Goddard (perteneciente al grupo llamado Bow Street Ronners) [8] [9]; él descubrió al asesino comparando la impresión o protuberancia que se encontraba en el proyectil dejado en el cuerpo de la victima con los moldes (mediante los que se fabricaban las balas) de los sospechosos; al comparar uno de ellos y fabricar una bala con el mismo descubrió que ésta y la incriminada eran idénticas y de esta manera encontró al culpable [10].

Posteriormente, en Neuruppin (Alemania) durante el año 1898, el Dr. Paul Jeserich, un médico forense berlinés, asistió un caso de homicidio. Se le facilitó el proyectil extraído del cuerpo de la víctima y el revólver perteneciente al acusado. Debido a su idea basada en que el proyectil sufría una serie de marcas al recorrer el ánima del cañón y rozar sus estrías a gran presión, decidió disparar con el arma del criminal para comparar el "proyectil testigo" con el "proyectil dubitado" y encontró una coincidencia exacta entre ambos; de esta forma pudo condenarse al asesino [11].

Después de varios dictámenes erróneos, Charles E. Waite (USA) comenzó a recolectar los datos exactos de las características de las armas fabricadas en su país y en Europa. A finales de 1923 concluyó que aun cuando las armas hicieran parte de la misma compañía de fabricación, su modelo las hacía distintas en algunos detalles: Calibres, número y orientación de estrías y ángulos de torsión; de esta forma, creó un catálogo técnico de las armas existentes en aquella época agrupando sus "características de clase" [12]. Sin embargo, estas características no eran suficientes para diferenciar armas pertenecientes al mismo modelo y, para solucionar este problema comenzó a observar el proceso de fabricación del cañón de una pistola; aquí descubrió que las máquinas usadas (a pesar de su precisión y calidad) dejan algunas

particularidades y con ello obtuvo información sobre las "características identificativas". No obstante, ellas requerían una observación en detalle, por lo que Waite le pidió al óptico Max Poser que fabricara un microscopio para verificarlas; el microscopio [13] construido constaba de un soporte que mantenía sujeta la bala y una escala de medición para medir las huellas más insignificantes que existieran en la misma. Tiempo después, se unieron a este estudio el físico John H. Fisher y el químico y especialista en microfotografía Philipp O. Gravelle y en conjunto formaron el primer instituto de balística forense en el mundo (Bureau of Forensic Ballistics), ubicado en New York [14].

Los estudios anteriores permitieron avances en el área de la balística; sin embargo, aún no existía la eficiencia necesaria para identificar las armas involucradas en un crimen de manera óptima y certera. Estas novedades en el reconocimiento de armas comenzaron a darse con la vinculación del procesamiento digital de imágenes. En el año 1999, durante una convención de tecnología de las ciencias forenses, se presentó la comparación de técnicas de reconocimiento de patrones y selección de características en armas de disparo implementando descriptores invariantes en la imagen y la transformada Wavelet [3], estos estudios concluyeron que aunque se encontró una buena respuesta en la clasificación, era necesario realizar proyectos futuros implementando otras técnicas. Los años posteriores sirvieron para encontrar nuevas estrategias para la identificación de armas; en 2009 un conjunto de investigadores, procedentes de China e India, realizaron este proceso de identificación por medio del Análisis de momentos geométricos en las 16 características obtendias del centro de la vainilla (donde se encuentra el cráter de percusión) [4], esto se hizo con una base de datos de 747 imágenes de vainillas para cinco diferentes armas, obteniéndose una exactitud de 74 % que les permitió concluir el potencial de ésta y otras técnicas relacionadas en el reconocimiento de armas de fuego. Para 2011, durante la conferencia de "Análisis de patrones e inteligencia robótica", se presentó nuevamente un estudio de identificación de armas para una base de datos con la misma cantidad de imágenes que en el anterior estudio, utilizaba esta vez las redes neuronales y específicamente su retropropagación (Backpropagation Neural Networks) [5]; esto se hizo por medio de redes con dos capas de retropropagación conectadas en secuencia 6-5-7 (además de capas internas computarizadas) y una función de transferencia sigmoide/lineal que, por medio de la clasificación "Cross-validation", obtuvo una exactitud de 96 % y permitió evidenciar que las redes neuronales son una excelente alternativa para el reconocmiento de armas. En 2012, como un aporte a la investigación de crímenes realizada por la policía de Malasia, se decidió estudiar la técnica de características numéricas en el cráter de percusión de las vainillas (usando la misma cantidad de imágenes que en las técnicas anteriores, de 5 pistolas diferentes provenientes de Sudáfrica) [6], obtuvieron 16 características ordenadas por orden de importancia y peso en la imágen a analizar, asignando aspi coeficientes para hallar el factor de sus componentes; validaron de nuevo con "Cross-validation" y obtuvieron una exactitud de 75,4 % que representó una gran utilidad para el cuerpo de investigación de la policía. Durante el año 2018, se retomó el estudio con BPNN (Backpropagation Neural Networks), utilizando la misma base de datos y por ello las mismas armas que en el estudio anterior, variando solo el orden de conexión de las dos capas (11-11-5) y algunas operaciones en las mismas (Max Pooling, Fully Connected Layers, entre otras) [15];

con la clasificación de este nuevo proceso, se obtuvo una exactitud de 87 % que demostró (aunque no con la eficacia del estudio de BPNN anterior) la utilidad de las redes neuronales para clasificación de imágenes balísticas.

2.2. PROCESAMIENTO DIGITAL DE IMÁGENES

El procesamiento digital de imágenes parte desde el fundamental teorema de Fourier. Sus principios de funcionamiento estaban establecidos desde hace muchos años; sin embargo, sólo hasta la creación de las computadoras pudieron estudiarse y aplicarse. La primera aplicación de este tipo de procesamiento fue hecha durante 1920, buscando reducir el tiempo para el envío de imágenes de prensa entre Londres y Nueva York, limitando el tiempo de una semana a tres horas. Lo anterior fue conseguido gracias al sistema de transmisión por cable *Bartlane*, que codificaba las imágenes en señales eléctricas que se reconstruían al otro lado de la línea telegráfica y se imprimían usando tipografías que simulaban un modelo de semitonos, primero con sólo 5 niveles de gradación, pero llegando a los 15 en el año 1929 [16].

Posteriormente, en 1957, Russell Kirsch, científico del National Institute of Standards and Technology creo la (hasta ahora reconocida) primera imagen digital, recreada en píxels. El científico utilizó un porto-escáner que transformó la imagen de su hijo pequeño en una matriz de ceros y unos, permitiendo visualizarla en el SEAC, único ordenador programable que existía entonces en EEUU y disponible precisamente en el NIST.

Sólo hasta que comenzó el interés por la exploración espacial y los avances médicos (década de 1960) empezaron a tratar de mejorarse las imágenes utilizando técnicas de manipulación de imágenes electrónicas mediante computadores. Sin embargo, no fue hasta la década de los años 70 que se produjo un gran avance en el procesamiento de imágenes; esto se debió a la creación de microprocesadores que permitieron ampliar la capacidad de almacenamiento y procesamiento de las imágenes. Los avances posteriores permitieron que, haciendo uso de computadores con procesadores más potentes, se realizaran grandes mejoras en el procesamiento digital de imágenes y que esto, aplicado a diversos campos, permitiera obtener resultados óptimos para la comparación y clasificación de las imágenes (como en este caso).

Para la clasificación correcta de las imágenes fue necesario crear, a partir del procesamiento digital de imágenes, técnicas que permitiera la extracción de características en las mismas [17]; dos de ellas fueron: Speeded Up Robust Features (SURF) y Convolutional Neural Networks (CNN).

2.2.1. SURF:

Este detector de características locales fue presentado por primera vez por Herbet Bay en 2006. Surgió con la idea de mejorar la eficiencia del descriptor SIFT (Scale-Invariant Feature Transform) [18] en los

siguientes aspectos:

- Velocidad de cálculo superior sin ocasionar pérdida del rendimiento.
- Mejor robustez ante posibles transformaciones de la imagen.

Consiguiéndose éstas mediante la reducción de la dimensionalidad y complejidad en el cálculo de los vectores de características de los puntos de interés obtenidos.

2.2.2. CNN:

Desde el siglo anterior se ha buscado una manera de comparar y clasificar las imágenes para distintos campos de la ciencia; esto sólo era posible manualmente, lo que acarreaba un prolongado tiempo y un considerable aumento de costo. Por tanto, se comenzó a buscar una técnica que realizara esta clasificación y comparación optimizando tiempo y recursos.

El primer acercamiento que se tuvo a las CNN fue dado en 1936 por el matemático y científico de la computación Alan Turing; éste comenzó a estudiar el cerebro como una forma de ver el mundo de la computación [19]. No obstante, sólo hasta 1943 se comenzaron a concebir los fundamentos de la computación neuronal de la mano de Warren McCulloch (neurofisiólogo) y Walter Pitts (Matemático); ellos lanzaron una teoría sobre la forma de trabajo de las neuronas [20], modelando una red neuronal simple mediante circuitos eléctricos. En 1949, Donald Hebb escribió el libro titulado: *La organización del comportamiento* [21], fundamental para el estudio de los procesos de aprendizaje que relacionan la psicología y la fisiología, siendo esto base para la inteligencia humana. Esta idea se basaba en que el aprendizaje se daba cuando se activaban ciertos cambios en una neurona y hallando semejanzas entre el aprendizaje y la actividad nerviosa. Los trabajos de Hebb formaron las bases de la teoría de Redes Neuronales [22]. En el siguiente año Karl Lashley encontró que la información no era almacenada de forma centralizada en el cerebro sino que era distribuida encima de él. Para el año 1956 el congreso de Dartmouth indicó el nacimiento de la inteligencia artificial.

Para 1957, gracias al algoritmo conocido como *perceptrón* y creado por Frank Rosenblatt [23], el cual recibía como elemento de entrada un vector y multiplicaba por un valor escalar a cada uno de sus componentes, este valor escalar se denominó *peso* (siendo distinto en cada componente). El algoritmo continuaba con la suma de todos los resultados y pasaba este valor final por una función de activación que podía plantearse de diversas maneras pero que tenía el principal objetivo de entregar una salida mayor de acuerdo con el valor de entrada (a mayor entrada, mayor salida), este procedimiento realizado con el fin de imitar la tasa de respuesta de las neuronas en el cerebro animal al recibir una estimulación.

La primera aplicación real de la Redes Neuronales, se dio en 1960 por medio de Bernard Widrow y Marcial Hoff y su modelo *Adaline* (ADAptative LINear Elements) [24], que buscaba mejorar la eficiencia

de los filtros adaptativos para eliminar ecos en las líneas telefónicas; siguió utilizándose durante varias décadas comercialmente. Karl Steinbeck desarrolló una red neuronal con simple memoria asociativa conocida como *Die Lernmatrix*. En 1967, Stephen Grossberg desarrolló una red llamada *Avalancha,* que consistía en el reconocimiento continuo de habla y aprendizaje de los brazos de un robot mediante ecuaciones diferenciales obtenidas a través de elementos discretos que variaban en el tiempo. Sin embargo, para esta técnica existió un periodo crítico causado por Marvin Minsky y Seymour Papert, esto debido a su publicación de un libro conocido como Perceptrons, en el que probaron matemáticamente que este tipo de algoritmo no era capaz de resolver problemas relativamente fáciles, como el aprendizaje de una función no lineal (funciones empleadas extensamente en el campo computacional). Sin embargo, esto no desmotivó a algunos de los investigadores que estaban haciendo su trabajo con esta técnica; un ejemplo fue James Anderson, que desarrolló un modelo lineal llamado *Asociador Lineal,* que consistía en elementos integradores lineales (neuronas) que sumaban sus entradas. El funcionamiento de este modelo se basaba en el principio de que cada vez que se activaban las neuronas, se reforzaban las conexiones entre ellas y, este investigador, desarrolló una extensión muy potente del modelo llamada *Brain State in a Box (BSB)* [25].

Para 1974, Paul Werbos desarrolló la idea básica del algoritmo de aprendizaje de propagación hacia atrás (Backpropagation) [5]; sin embargo, su significado sólo fue aclarado totalmente hasta 1985. Nuevamente, en 1977, Stephen Grosberg realizó la arquitectura de red conocida como Teoría de Resonancia y Adaptada (TRA), diferenciada de las inventadas anteriormente, además de tener la capacidad de memoria a corto y largo plazo. En 1980, Kunihiko Fukushima desarrolló un modelo neuronal para el reconocimiento de patrones visuales. Para 1986, David Rumelhart redescubrió el algoritmo de propagación hacia atrás; a partir de ese año comenzó a indagarse y realizarse más investigaciones acerca de Redes Neuronales, llevándose todavía a cabo aplicaciones en diversas áreas (sobretodo en control y mercadeo), viéndose un uso destacado en el reconocimiento de armas [26].

Capítulo 3

MARCO TEÓRICO

3.1. BALÍSTICA FORENSE

Se define como “la ciencia que estudia todos los fenómenos relacionados con el comportamiento del proyectil de un arma de fuego, desde el momento del disparo y hasta su llegada al punto de impacto.” [27]. La balística forense puede ser estudiada teórica y prácticamente, la primera ayuda a determinar el arma desde la que se disparó un proyectil, mediante el estudios comparativos, metodologías o cotejos. La siguiente forma de estudio se encarga de recrear escenas de crímenes con el fin dar solución a casos e investigaciones de los mismos. La balística forense puede clasificarse en interior, exterior, de efectos e identificativa[28].

La balística interior se refiere a la parte que se encarga del estudio de todos los fenómenos que se producen en el arma desde el momento en que el percutor golpea el fulminante del cartucho hasta el momento mismo en que el proyectil abandona la boca de fuego del cañón. Esta parte de la balística se ocupa también de todo lo competente a las armas de fuego, su estructura, mecanismos, funcionamiento, carga y disparo de las mismas [29].

Por otro lado, la balística exterior es la parte de la balística a la cual le corresponde el estudio de la trayectoria del proyectil, desde el momento en que abandona la boca del cañón del arma hasta su llegada al objetivo, y de los fenómenos que lo afectan en concordancia con las particularidades de cada caso, tales como la gravedad, la resistencia del aire, la influencia de la dirección e intensidad de los vientos y particularmente los obstáculos que se le interpongan y que en definitiva son productores de los rebotes que modifican la trayectoria original [30].

La balística de efectos, tal como su nombre lo indica, estudia los efectos producidos por el proyectil en el blanco alcanzado, particularmente las características propias del Orificio de Entrada (OE) causado por el proyectil y de la zona inmediata que lo rodea [31], estas características proporcionan importantes

elementos mediante los que se extraen conclusiones relativas a problemas tan complejos como la determinación de la distancia de disparo.

La balística identificativa por su parte es la rama de la balística se encarga de analizar las coincidencias o similitudes entres las huellas dejadas, en el momento del disparo, al proyectil o vainilla, con las características de un arma determinada. Estas huellas son dejadas por el contacto entre partes del arma, tales como, las irregularidades al interior del cañón del arma, la aguja percutora y por la contra-recamara y la munición. Por medio de estas características, un experto en balística puede determinar por qué tipo de arma fue percutida la vainilla [32]. Se tienen en cuenta distintos tipos de características.

- C**aracterísticas generales del estriado:** hacen referencia al número de estrías y macizos que presenta un cañón del arma de fuego de determinado calibre, así como su ancho y sentido de giro.
- C**aracterísticas de clase:** son aquellas que se comparten entre varias armas y se dan en el momento de diseño. Éstas pueden ser: marca, modelo, calibre, entre otras.
- **Características de subclase:** son aquellas que resultan de incidencias en los procesos de fabricación; por tanto, son más significativas que las anteriores, ya que cada lote de fabricación será distinto de los demás. Uno de los factores que intervienen en la aparición de estas características es el tiempo, puesto que los procesos de elaboración de las armas varían según el paso de éste. [33]
- **Características individuales:** son aquellas que se producen accidentalmente al momento de la elaboración de un arma en particular o que se derivan de la corrosión o el daño de la misma por su uso. Estas características son exclusivas de cada arma y permiten el mayor nivel de identificación. Un ejemplo puede ser particularidades en el interior del cañón de un arma de fuego de ánima rayada. [34]

3.1.1. Armas de Fuego

Son aquellas que utilizan la energía de los gases producidos por la detonación de pólvora con el fin de lanzar un elemento sólido (que es generalmente metálico) conocido como proyectil a cierta distancia y hacia un objeto (blanco) determinado [35]. Entre los distintos tipos se encuentra el revólver, que es un arma de fuego de corto alcance, con un funcionamiento mecánico en el cual, por su mecanismo de disparo puede ser de simple acción, en la que para concretar un disparo es necesario previamente montar el martillo (desplazarlo totalmente hacia atrás de forma manual) o de doble acción. En este último caso el disparo se logra oprimiendo directamente la cola del disparador. También se cuenta con la pistola (ver Figura 3.1), se trata de un arma generalmente semiautomática, que no precisa demás acción mecánica por parte del tirador que desplazar debidamente la corredera, y las sucesivas presiones del dedo índice sobre la cola del disparador harán posibles los correspondientes disparos. Otro de sus ejemplares es la escopeta, descrita como un arma larga o de hombro, de uno o dos cañones de ánima lisa, en la que normalmente

se emplean cartuchos de perdigones. En la siguiente imagen se muestra una de las pistolas con las que se percutieron las vainilas utilizadas en el proyecto; esta fue facilitada por la Fiscalía seccional Pereira.

Figura 3.1: Pistola Sig Sauer, modelo SP2022, 9 mm.

Para la clasificación de estas armas, muchas veces se usa el calibre; éste es la media del diámetro interior de su cañón, es decir, el limitado por las paredes constitutivas de su ánima, denominándose así el hueco del cañón o tubo metálico característico de toda arma de fuego, comprendido entre el extremo cerrado y la abertura que da al exterior, denominada boca de fuego, la cual conforma el calibre propiamente dicho, dado que el ánima se compone o divide en dos partes: la recámara y el ánima rayada, siendo la primera de ellas la que sirve de alojamiento a la vaina del cartucho.

3.1.2. Cartucho

Se denomina así al conjunto de elementos que son utilizados en el arma para generar el impacto; todos éstos se encuentran contenidos en la vainilla [35]. Éste se divide en varias partes o componentes:

- O**jiva:** elemento hueco, cónico y metálico diseñado para cubrir la parte delantera de un proyectil, y reducir la resistencia del aire durante el vuelo.
- P**royectil:** es un elemento con punta afilada o ergonómica que tiene como finalidad impactar en un blanco u objeto.
- V**ainilla:** su labor es contener todos los elementos del cartucho. Al momento del disparo (y según el tipo de arma) ésta puede caer o quedar en el interior del arma para ser removida manualmente. Y sus partes esenciales son: Boca, cuerpo, ranura y culote o base. La vainilla tiene entre sus

componentes a la boca, que es la parte abierta de la vainilla que sirve para enlazar la bala. Otra de sus partes es el cuerpo, Son las paredes cilíndricas en las que está contenida la pólvora y parte del proyectil. Estas paredes son más gruesas cerca del culote, en donde son sometidas a mayor presión. También se encuentra el culote, que hace referencia a la base de la vainilla que tiene una forma plana en su parte exterior y una pestaña o ranura para que sea posible extraer la vainilla del arma. Si el cartucho es de percusión central, el culote tiene un orificio en su centro para alojar la cápsula fulminante, siendo el culote de mayor grosor que el cuerpo de la vaina. Si el cartucho es de percusión anular, el grosor del culote es fino y no lleva cápsula fulminante, ya que el fulminante se encuentra dispuesto en el interior del reborde del culote, siguiendo la periferia de la base del cartucho. En este caso, el percutor del arma, en vez de golpear la cápsula en el centro debe golpear cualquier punto de la periferia para producir la ignición. Existe además la pólvora, que define a las mezclas o combinaciones químicas encargadas de generar por su combustión los gases que impulsan el proyectil.

3.2. PRELIMINARES DEL PROCESAMIENTO DIGITAL DE IMÁGENES

Para este tipo de procesamiento es necesario tener presentes conceptos como imagen, que se define matemáticamente como una función con coordenadas espaciales bidimensionales (x,y) en un plano; es útil específicamente la definición de imagen digital, que es aquella imagen que posee cantidades finitas y discretas para sus coordenadas x, y; se compone de un número finito de elementos con valor y lugar específicos conocidos como píxels.También hace parte de estos conceptos la visión por computador, descrita como un conjunto de herramientas y métodos que permiten la obtención, procesamiento y análisis de imágenes del mundo real para tratarlas computacionalmente y automatizar ciertas tareas para la obtención de información y posterior toma de decisiones [36].

3.3. PROCESAMIENTO DIGITAL DE IMÁGENES (PDI)

Es un conjunto de técnicas aplicado a imágenes digitales buscando mejorar su calidad y facilitando la obtención de información a partir de las mismas. Existen cuatro etapas que comienzan en el área de procesamiento digital de imágenes y finalizan en el área de visión por computador [37]:

- **Preprocesamiento digital de imágenes:** este conjunto de técnicas busca mejorar características en la imagen; para lograrlo, puede adecuarse el tamaño, reducirse el ruido, mejorar el contraste y el enfoque. El resultado será una imagen óptima para extraer características que brinden información útil para la situación a estudiar.

- **Segmentación**: consiste en realizar una partición de la imagen en secciones significativas (de acuer-

do con la situación o el caso con el que se esté trabajando). Estos métodos asumen que cada una de las regiones, sobre las que la imagen ha sido dividida, cuenta con algunas características homogéneas distintivas y estos pueden percibirse como procesos de reconocimiento de patrones o como procesos de decisión. Los resultados obtenidos pueden ser atributos extraídos de la imagen, tales como contornos y bordes.

- **Representación y descripción:** esta etapa hace uso de algoritmos basados en el reconocimiento de bordes y/o propiedades internas de la imagen para la descripción de las regiones de interés.

- **Reconocimiento e interpretación:** este proceso consiste en etiquetar un objeto basándose en la información suministrada por sus descriptores y asignar un significado al conjunto de objetos reconocidos para una correcta interpretación de los resultados.

3.4. SPEEDED UP ROBUST FEATURES (SURF)

Es un algoritmo comúnmente utilizado para la extracción de puntos de interés en el reconocimiento de imágenes. Esta extracción se realiza detectando los posibles puntos de interés y su localización en la imagen. Éste se basa en la técnica SIFT (Scale-Invariant Feature Transmorm), que es una transformación de información proporcionada por una imagen en coordenadas invariantes a la escala en el ámbito local [38]. La diferencia entre ambas técnicas radica en que SURF implementó mejoras al algoritmo trabajado por SIFT; entre ellas se encuentran:

1. Mayor velocidad de cálculo sin generar una pérdida en el rendimiento.

2. Mayor robustez ante posibles transformaciones de la imagen.

La rapidez de SURF respecto la técnica SIFT tiene que ver con el uso de un menor número de descriptores (ya que som iguales a cero en su mayoría) en cada uno de los keypoints. Lo anterior no representa una alteración drástica en la codificación del algoritmo, ya que siguen usándose gran cantidad de funciones implementadas en el algoritmo de la técnica SIFT. Esta técnica consta de tres pasos.

3.4.1. Detección de Puntos de Interés

Esta etapa hace uso del valor del determinante de la matriz Hessiana para la localización y escala de los puntos; ésta es importante porque brinda respaldo en cuanto a la velocidad de cálculo y la precisión de los mismos.El descriptor SURF incluye en su detector algo novedoso, esto es la utilización del valor del determinante de la matriz Hessiana para el cálculo de la posición y escala de lo puntos de interés; a diferencia de los detectores de otros descriptores que, buscando calcular esta posición y escala, utilizan diferentes medidas para los puntos individualmente [39].

3.4.2. Asignación de la Orientación

En esta etapa se otorga la invarianza mediante su rotación a cada uno de los puntos de interés obtenidos en la etapa anterior mediante su orientación. Para esto, deben realizarse cálculos de la respuesta de Haar en ambas direcciones (x,y). Posteriormente se realiza el muestreo, el cual depende de la escala (a mayor escala, mayor dimensión de las funciones). Finalmente, se usan imágenes integrales para realizar su filtrado por medio de máscaras de Haar y obtener de esta forma su respuesta en ambas direcciones.

3.4.3. Descriptores SURF

El primer paso es construir una región cuadrada alrededor de los puntos de interés, con orientación en relación a la calculada en la etapa anterior; ésta se divide en subregiones con el fin de obtener las respuestas de Haar para puntos con una distancia límite de separación en ambas direcciones. Para hacer los valores de respuesta más invariantes a deformaciones geométricas y errores de posición, éstos se ponderan con una gaussiana centrada en el punto de interés; en cada subregión se suman los valores anteriormente mencionados y de esta forma se obtiene un valor representativo para ellas individualmente. Al tomarse las subregiones de manera general, se genera un descriptor SURF con 64 valores (cada uno representando una característica) en cada punto de interés previamente identificado [40].

3.5. CONVOLUTIONAL NEURAL NETWORKS (CNN)

Las redes neuronales en general, se conocen como un modelo computacional y simplificado del cerebro humano, con la capacidad de adquirir conocimiento a través de la experiencia. Podría expresarse entonces que una red neuronal es: "Un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: *La neurona*" *[41].*

Concretamente, las redes neuronales convolucionales funcionan de forma similar a una red neuronal ordinaria; dando a las neuronas de las que se compone pesos y sesgos que pueden aprender. A cada una de las neuronas se le ingresa una entrada, se realiza un producto escalar y se aplica finalmente una función de activación. La diferencia entre una red neuronal ordinaria y una red neuronal convolucional es que las entradas de éstas últimas son imágenes y presentan mayor eficiencia frente a imágenes de alta definición. Su funcionamiento puede explicarse a grandes rasgos de la siguiente manera: Con la primera capa se trata dedetectar los bordes y establecer patrones para esta detección. A continuación, las capas posteriores intentan combinarlos en formas más simples y, finalmente, en patrones de las diferentes posiciones de los efectos, iluminación, escalas, etc. Las capas finales tratan de hacer que coincida una imagen de entrada con todos los patrones y arroja una predicción final como una suma ponderada de todos ellos [42]. Generalmente, este tipo de redes constan de una estructura que contiene tres tipos de capas diferentes:

3.5.1. Capa Convolucional

Recibe este nombre gracias a la operación que allí se realiza (convolución). Ésta consiste en recibir la imagen a procesar como una entrada y aplicar sobre ella un filtro o kernel que entrega un mapa de las características de la imagen original, reduciendo de esta manera el tamaño de los parámetros. Esta operación hace uso de tres ideas importantes (útiles en cualquier sistema de Machine Learning):

- **Interacciones dispersas:** al aplicar un filtro de menor tamaño sobre la entrada original, puede reducirse considerablemente la cantidad de parámetros y cálculos.
- **Parámetros compartidos:** al compartirse éstos entre distintos tipos de filtros se mejora la eficiencia del sistema.
- **Representaciones equivariantes:** indica que al varias las entradas, variarán también las salidas en forma similar.

Además, en este punto se utiliza la función de activación ReLU (Rectified Linear Unit), cuyo fin es conectar la salida de la neurona anterior con la entrada de la neurona presente, asignando un valor a la misma (cuidando además la salida presentada sea lineal). Esta función de activación en particular, presenta beneficios respecto a las demás, entre ellos: conserva el valor de las derivadas de las capas al multiplicarlas por uno durante la etapa de backpropagation, optimiza el tiempo de entrenamiento mediante la reducción de costo computacional y finalmente brinda mejores resultados al imitar el comportamiento del cerebro humano [43].

3.5.2. Capa de Reducción o Pooling

Ésta se ubica generalmente seguida de la capa convolucional. Es importante porque permite la reducción de las dimensiones espaciales (ancho x alto) del volumen de entrada para la siguiente capa convolucional; es importante especificar que la profundidad del volumen no se verá afectada. La operación realizada en esta capa se conoce como *reducción de muestreo [44],* ya que la reducción de tamaño conduce también a la pérdida de información. Sin embargo, esta pérdida no representa un problema por las siguientes razones:

- La disminución en el tamaño genera una menor sobrecarga de cálculo para las próximas capas de la red.
- Con esta pérdida se reduce también el sobreajuste.

Además suele utilizar una operación llamada *max-pooling,* que divide a la imagen de entrada en un conjunto de rectángulos y, respecto a cada subregión, toma el máximo valor.

3.5.3. Capa Clasificadora Totalmente Conectada

Después de aplicar las capas anteriormente mencionadas, las redes generalmente utilizan capas completamente conectadas, en las que cada pixel se considera como una neurona separada. Esta última capa posee tantas neuronas como el número de clases que debe predecirse [45].

Las redes neuronales convolcionales contienen una serie de parámetros tales como **Kernel**, que para el caso de las CNN puede ser tomado como filtro o mapa de características; mediante él y sus pesos, se definen las características que serán tomadas para la siguiente capa de convolución. Las dimensiones del Kernel son dadas según la necesidad que se tenga; es decir, pueden variarse con el fin de encontrar el valor que provea una mejor exactitud en la etapa final de las redes; otro de sus parámetros es el **Número de Filtros**, el cual hace referencia al número de neuronas que componen una capa convolucional, determinando el tamaño de la capa de salida a partir de ésta. Al igual que el parámetro anterior, el número de filtros puede ser variado con el fin de encontrar una mayor exactitud en la etapa final de las redes; sin embargo, un número elevado de filtros genera un costo computacional más alto. También se encuentra el **Stride**, cuya definición es el salto entre los pixeles de la imagen para crear los valores de la nueva capa. El valor del Stride definirá la cantidad de pixeles que se saltan para establecer el valor de la nueva casilla en la capa de salida. Tiene por tanto, dimensiones en el eje x y el y; estos definen las casillas saltadas horizontal y verticalmente. Una manera de definir el valor del Stride es intentar obtener un número entero a la salida de la capa (ya que las imágenes no contienen dimensiones fraccionarias). Por último se presenta el **Padding**, este parámetro consiste en el relleno de los bordes de la imagen con ceros y, su importancia radica en la capacidad de obtener una imagen a la salida de las capas de convolución con las mismas dimensiones de la imagen de entrada. Cada capa de convolución disminuye el tamaño de la imagen y con ella se pierde información importante (sobre todo si se analizan los bordes de la imagen); el relleno de ceros con este parámetro, permite mantener la información valiosa en estos sectores de la imagen; el valor de Padding depende de la cantidad de relleno en los bordes que se requiera para la imagen de salida, pudiendo variar estos valores para la parte izquierda, derecha, superior e inferior de la misma.

Si se desea calcular las dimensiones para las capas de agrupación y la imagen resultante de la red neuronal convolucional, es necesario aplicar la ecuación 3.1 (que hace uso de los parámetros anteriormente mencionados):

$$Convolución = \left(\frac{W - F + 2 * P}{S} + 1 \right) \tag{3.1}$$

Donde:

- **W:** Dimensión de la imagen.

- **F:** Dimensión del kernel.
- **P:** Valor de Padding.
- **S:** Valor de Stride.

Las redes neuronales convolucionales además cuentan con otras variables que permiten mejorar su exactitud en los resultados de salida de la imagen; entre éstas se encuentra las **Epoch** (épocas) [46], que se define como una medida del número de veces que todos los vectores de entrenamiento son usados para actualizar el peso de las neuronas. Para los lotes de entrenamiento, todas las imágenes pasan a través del algoritmo de aprendizaje de manera simultánea en una época (epoch) antes que sus pesos sean actualizados. También se encuentra el **Batch**, que se conoce como el número de muestras que pueden ser propagadas a través de una red neuronal (la división de estas muestras en cantidades más pequeñas es llamada Mini-Batch). Finalmente la variable **Iteración** describe el número de veces que un lote de datos pasa a través del algoritmo.

3.6. CLASIFICACIÓN

Para esta etapa, se hace uso de la matriz de confusión [47]; ésta no es más que una matriz que muestra los resultados obtenidos en un proceso de validación ubicando los valores reales horizontalmente y los predichos verticalmente. Esto sirve para identificar los siguientes parámetros:

- Verdaderos positivos (*True Positive*): Aquellos valores que son clasificados por un algoritmo como positivos y que coinciden con el valor real.
- Verdaderos negativos (*True Negative*): Aquellos valores que son clasificados por un algoritmo como negativos y que coinciden con el valor real.
- Falsos positivos (*False Positive*): Aquellos valores que son clasificados por el algoritmo como positivos pero en realidad debieron clasificarse como negativos.
- Falsos Negativos (*False Negative*): Aquellos valores que son clasificados por el algoritmo como negativos pero en realidad debieron clasificarse como positivos.

A partir de lo anterior, habiendo construido la matriz, se obtienen los resultados de *Exactitud* (ver Ecuación 3.2), que define la cantidad de valores clasificados correctamente entre todos los valores posibles; el *Error* (ver Ecuación 3.3), que define la cantidad de predicciones obtenidas que son incorrectas; la *Sensibilidad* (ver Ecuación 3.4), que define como la capacidad del algoritmo para predecir valores positivos que realmente lo son; por último, la *Especificidad* (ver Ecuación 3.5) que se define como la capacidad del algoritmo para predecir valores negativos que realmente lo son.

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

$$Error = 1 - AC \tag{3.3}$$

$$TPR = \frac{TP}{TP + FN} \tag{3.4}$$

$$TNR = \frac{TN}{TN + FP} \tag{3.5}$$

Capítulo 4

METODOLOGÍA

4.1. CREACIÓN DE BASE DE DATOS

Para la recolección de las imágenes que harían parte de la base de datos se contó con un convenio entre la Universidad Tecnológica de Pereira y la Fiscalía General Seccional Pereira, quienes por medio de asesores como el investigador Jaime Granada Hincapié y demás ingenieros, facilitaron, además de la muestra para esta base (250 vainillas pertenecientes a un total de cinco diferentes armas, concretamente pistolas Sig Sauer modelo SP2022 y calibre 9 mm, como se muestra en la Figura 3.1), los instrumentos para la captación de las imágenes requeridas por la muestra; éstos fueron el estereoscopio digital focal Leica modelo M205A y el software Leica Application Suite (ver Figura 8.6). El procedimiento efectuado para esta etapa comenzó con la marcación de las vainillas según el número de caja a la que pertenecían (indicando cada caja el arma) y su posición en dicha caja entre 50 posibles. Después de esto, y antes de comenzar la toma de imágenes para la base de datos, se verificó que la base de la vainilla se encontrara libre de impurezas (siempre y cuando no se tratara de impurezas por oxidación, casos en los que no pudo mejorarse su condición de limpieza), para removerlas, se aplicó Acetona a un copito y se repasó la superficie con el mismo; posteriormente, se ubicó la vainilla en la platina del estereoscopio colocando la huella del eyector en la parte horizontal derecha (conocida como 3 en la posición de horas del reloj). Al haberse realizado lo anterior, se procedió a captar las imágenes de la base, el fulminante y el eyector de la vainilla bajo distintos tipos de luz (nombrados de igual manera que las horas del reloj) y con aumento y coordenadas que permitieran enfocar de la mejor manera las características que ésta posee. El software cuenta con la herramienta multifoco para la captación de las imágenes; esto consiste en la toma de 13 fotografías desde el fondo del cráter de percusión hasta la superficie metálica de la vainilla, éstas son integradas en una sola imagen que, por tanto, contiene mayores detalles que una imagen capturada de forma normal; esto implicó que las imágenes resultantes fueran de una muy alta resolución (del orden de los 1579 x 2121), ocupando mayor espacio de almacenamiento (el formato de las imágenes fue TIF). Para guardar estas imágenes, las mismas fueron catalogadas referenciando la parte de la vainilla a la que pertenecían (base, fulminante y eyector), el valor de aumento establecido y las luces utilizadas en

la toma mediante la lámpara del estereoscopio (ver Figura 8.7). Se contó con un total de diez imágenes por vainilla, nombradas de la manera mostrada en la tabla de anexos (ver Tabla 8.1). El procedimiento anterior se realizó para las 50 vainillas que componen cada una de las cinco cajas; esto indica un total de 250 imágenes que se tomaron como muestra para la creación de la base de datos. El almacenamiento de las mismas, mediante un disco duro, fue ordenado por medio de carpetas y subcarpetas que refieren las cajas y las vainillas respectivamente.

4.2. ETAPA DE PRE-PROCESAMIENTO

Al analizar las imágenes con las que se iba a realizar el procesamiento, pudo evidenciarse que la región de interés se encuentran en las imágenes nombradas “B”, que contuvieron el centro del fulminante (ver Figura 4.1).

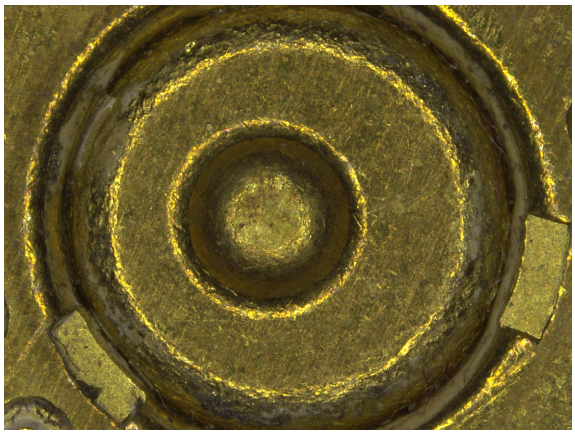

Figura 4.1: Fulminante de la vainilla con iluminación C0.

Como primera medida, para trabajar con imágenes que mostraran eficazmente las características a evaluar mediante las técnicas de extracción, se realizó la binarización y reducción de tamaño de las mismas de modo que pudieran ser procesadas con mayor facilidad mediante el programa de cálculo numérico MATLAB 2018 (ver Figura 4.2).

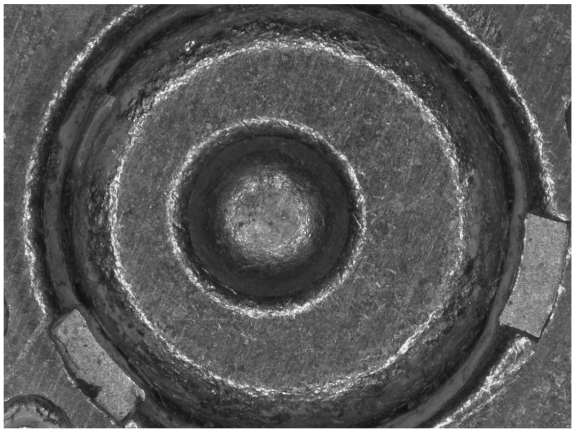

Figura 4.2: Fulminante recortado y binarizado.

Sin embargo, este tipo de imágenes aún contenían un gran número de características que habrían entorpecido la detección de puntos de interés en esta región particular y generarían mayor costo computacional a la hora de compilar los algoritmos; buscando obtener mejores resultados y mayor exactitud durante la clasificación mediante minería de datos, se llevó a cabo el pre-procesamiento de las imágenes mediante la umbralización y segmentación de las mismas, utilizando las medidas del radio (encontradas a prueba y error) de la circunferencia generada en el centro del fulminante para resaltar y posteriormente recortar la región de interés en la que se contiene la información deseada para la extracción de características utilizando el detector de bordes "Canny" (que proporcionó buena detección, buena localización y respuesta mínima para que el ruido no generara falsos bordes) y segmentando la imagen obtenida por medio de la transformada de Hough, que permitió detectar las curvas (fulminante-cráter de percusión) en las imágenes mediante funciones establecidas en Matlab 2018 (ver Figuras 4.3, 4.4 y 4.5). Finalmente, éstas fueron almacenadas en una variable citada en el algoritmo.

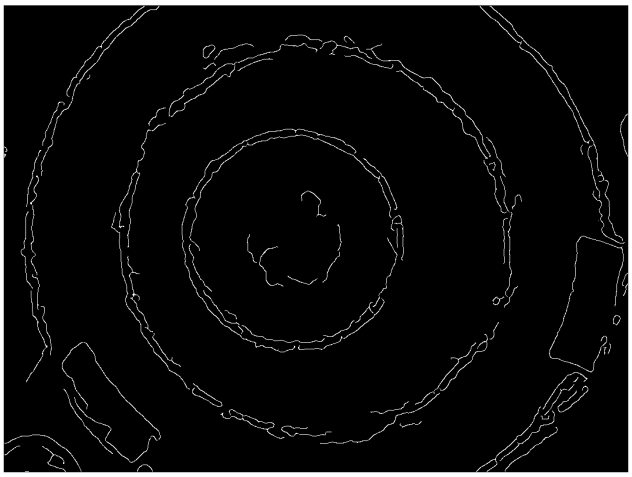

Figura 4.3: Fulminante después de su detección de bordes.

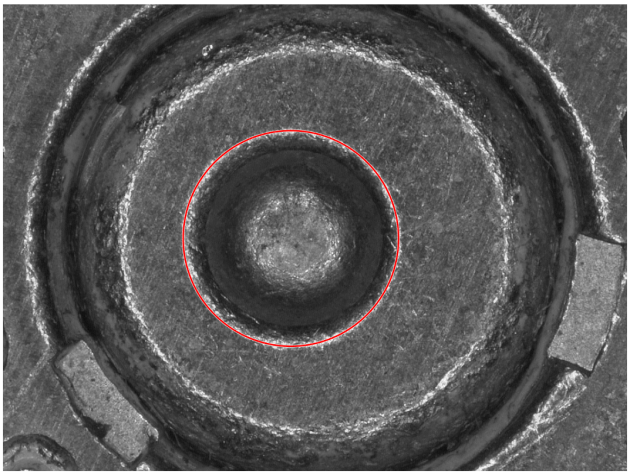

Figura 4.4: Diámetro central del fulminante.

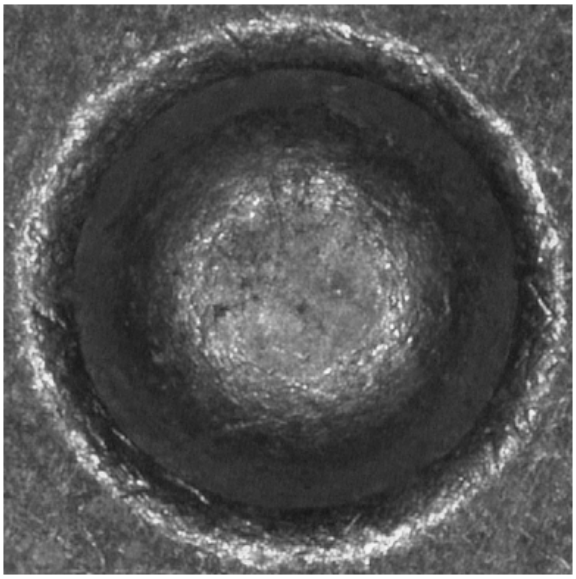

Figura 4.5: Fulminante segmentado mediante transformada de Hough.

4.3. ETAPA DE PROCESAMIENTO

4.3.1. Speeded Up Robust Features (SURF):

Las imágenes almacenadas en la variable final (mencionada en la etapa anterior) fueron procesadas bajo el algoritmo usado para la técnica SURF con el fin de conocer la localización de los puntos en los que se encontraron las 64 características detectadas de forma consistente por esta técnica para cada una de las 250 imágenes que conforman la base de datos. A partir de la carga de cada imagen en el algoritmo SURF, se obtuvo un vector que contuvo los puntos solicitados, siendo éstos representados con una cruz encerrada por un círculo (el cual variaba de tamaño según el peso del punto de interés: a mayor peso,

mayor diámetro del círculo) tal como se muestra en la figura 4.6; sin embargo, es importante aclarar que la cantidad de puntos varió de una imagen a otra y que, debido a que la finalidad de la recolección de estos puntos era la creación de una nueva base de datos (buscando contener en esta ocasión valores numéricos en vez de imágenes) en la que se agrupara los puntos obtenidos para las 64 características en cada una de las vainillas y posteriormente entrenarla y probarla con diferentes clasificadores de la minería de datos, cada vainilla o imagen debía poseer la misma cantidad de puntos y que éstos además debían ser los de mayor importancia; por esta razón, se redujo el número de puntos a obtener de acuerdo al menor número que se presentó durante la carga de la totalidad de imágenes y se introdujo una línea de código especializado en detectar los puntos de las características con mayor peso; se halló que el menor número fue 82 y que por tantos las dimensiones del vector, conformante de la base de datos de valores, sería de 82x64 para cada imagen. Al finalizar la carga de imágenes, se obtuvo una base de datos con las dimensiones 20418x64. Después de un proceso de entrenamiento y prueba (descrito con detalle en la etapa de clasificación) se constató que los 82 puntos en su totalidad podrían estar afectando la exactitud de clasificación de la técnica y que una reducción de cantidad de los mismos podía ser viable argumentando que algunos de los puntos, a pesar de ser catalogados como los de mayor peso, estarían brindando información no requerida acerca de las características de la vainilla. Frente a esto, se redujo la cantidad de puntos a 70 y se continuaron eliminando gradualmente 10 hasta contar con un vector de 10x64, registrándose los resultados con cada una de estas reducciones (ver Figura 4.6). La línea de código mencionada anteriormente permitió evidenciar que el vector obtenido, con cualquier cantidad de puntos, vino ordenado con sus componentes de mayor a menor peso.

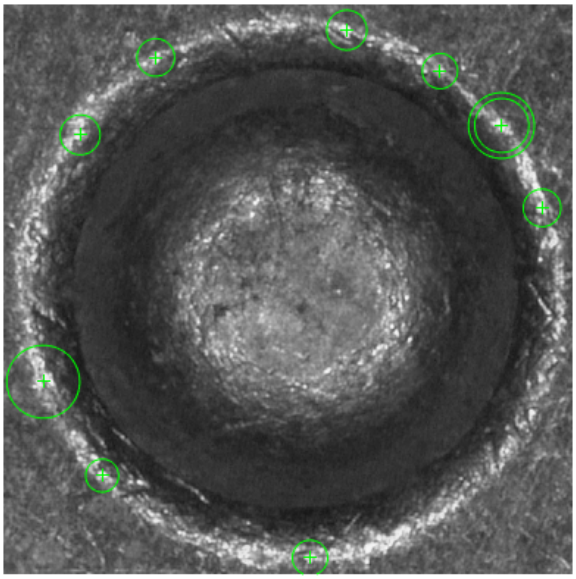

Figura 4.6: Puntos de interés en la imagen segmentada.

Etapa de clasificación:

Se estipuló para la etapa de clasificación que la base de datos debía dividirse de la siguiente manera: 70 % para entrenamiento y 30 % para prueba; por tanto, debieron crearse dos nuevas bases de datos (entrenamiento y prueba), para el entrenamiento debió anexarse una última columna que contenía la clase para cada vainilla (caja a la que pertenecía). Mediante el software de análisis numérico de MATLAB se evaluó la base de datos de entrenamiento con los valores de localización de los puntos de interés. Este software contiene un menú que permite seleccionar el clasificador a utilizar entre distintos tipos posibles; para el tema a evaluar se decidió trabajar con K-Nearest Neighbors (KNN) y con Support Vector Machine (SVM). Para el primer clasificador se usaron las distancias Euclidean, City Block y Mahalanobis (ver Ecuaciones 4.1, 4.2 y 4.3).

$$D = \sqrt{\sum_{i=1}^{n} (A-B)^T (A-B)} \tag{4.1}$$

$$D = \sum_{i=1}^{n} |a_i - b_i| \tag{4.2}$$

$$D = \sqrt{\sum_{i=1}^{n} (A-B)^T S^{-1} (A-B)} \tag{4.3}$$

El segundo clasificador hace uso de un elemento conocido como Kernel, que varía su valor según la clase utilizada entre Fine SVM, Medium SVM y Coarse SVM. Después de ejecutar la aplicación ésta generó un código que fue exportado como una función de MATLAB; de manera conjunta se creó un algoritmo que permitiera cargar la base de datos de prueba y predecir, por medio de la función anteriormente mencionada, la clase que correspondía a cada vainilla. El valor de clase que fue entregado para cada imagen se almacenó en un vector con dimensiones 6068x64. Por medio del programa Excel se guardaron estos valores y se realizó el conteo de las vainillas predichas en cada una de las cinco clases. Finalmente, y para obtener la exactitud de esta técnica aplicada a este problema en particular, se realizó la sumatoria de los conteos mencionados anteriormente y se creó la matriz de confusión a partir de la misma (ver figura 5.1). En esta matriz se confrontaron las clases predichas y las reales y, a partir de esta relación, se calculó la exactitud del procedimiento.

4.3.2. Convolutional Neural Networks (CNN)

Para la técnica de redes neuronales, las imágenes fueron transformadas a escala de grises y recortadas con el fin de que todas tuvieran el mismo tamaño (tomando como referencia el menor: 1579 x 2121). Posteriormente, se diseñó un protocolo que contuvo las opciones de variación a diferentes parámetros; esto permitió que el proceso de entrenamiento de la red convolucional se realizara de manera ordenada y con un número determinados a variar. Los pasos contenidos en este protocolo fueron 5 y se describirán a

continuación: En primera instancia se estableció el número de imágenes con el que quería trabajarse, para esto, se creó un algoritmo que dependiendo del número de imágenes seleccionado buscaba este criterio por cada clase en la base de datos de manera aleatoria, teniendo en cuenta dos imágenes más para la etapa de validación; con lo anterior, se obtuvo la siguiente información:

- Tamaño de la imagen.
- Imágenes por cada clase.
- Clases.
- Total imágenes.

Se definió la cifra total de imágenes a partir de la multiplicación del valor escogido con la cantidad de clases (siendo 5 en este caso por ser las armas totales a reconocer). Fue necesario calcular el tamaño de la imagen, teniendo en cuenta que la carpeta de trabajo (*Current Folder*) en Matlab debía contener las imágenes en carpetas con el nombre de su respectiva clase (definir su ruta para extraer la información de forma matricial).

Seguidamente, se cargaron las imágenes en una base de datos; esto se hizo creando matrices 4D definidas para imágenes de entrenamiento (*X_train*) e imágenes de validación (*X_test*), con sus respectivas dimensiones (ancho x alto x profundidad x índice de imagen, esta última representaba la posición de la misma en la carpeta a la que pertenecía) con el número de imágenes definido en el paso anterior, siendo seleccionadas de manera aleatoria. Para cada imagen se tuvo una categoría correspondiente a la clase a la que pertenecía, dichas categorías se guardaron en un vector para entrenamiento (*Y_train*) y un vector para prueba (*Y_test*) usando herramientas categóricas propias del Matlab. (Debía descartarse siempre la primera imagen de cada carpeta pues su índice generaba conflictos en el código que interrumpían el proceso de compilación, Esto se corrigió permitiendo la selección aleatoria sólo a partir de la segunda imagen en cada carpeta, no se conoce el motivo de esta limitación, pero se evidenció que no sucedía en el sistema operativo Windows 10 de 64 bits, por lo que la línea de código puede descartarse cuando el algoritmo es compilado en este tipo de equipos). Adicional a esto, se creó un vector de categorías para discriminar las clases por números (para el caso, estas categorías iban de 0 a 4 por ser el total de armas a reconocer); también se definió un índice que tomó la penúltima y última imagen de cada clase con el fin de establecerlas como imágenes de validación (almacenadas en una nueva variable). Después de guardar la base de datos, se creó una matriz de imágenes de prueba y un vector de categoría para éstas. A continuación, se muestra la información:

- Imágenes para entrenamiento: (*X_train*) = 1579 x 2121 x 1 x número_de_imágenes_a_entrenar.
- Clases imágenes entrenamiento: (*Y_train*) = número_de_imágenes_a_entrenar x 1.
- Imágenes para validación: (*X_test*) = 1579 x 2121 x 1 x número_de_imágenes_a_validar.

- Clases imágenes validación: (*Y_test*) = número_de_imágenes_a_validar x 1.

A continuación, y teniendo en cuenta lo anterior, se muestra la representación de las matrices 4D y los vectores de categorías (ver Figura 4.7):

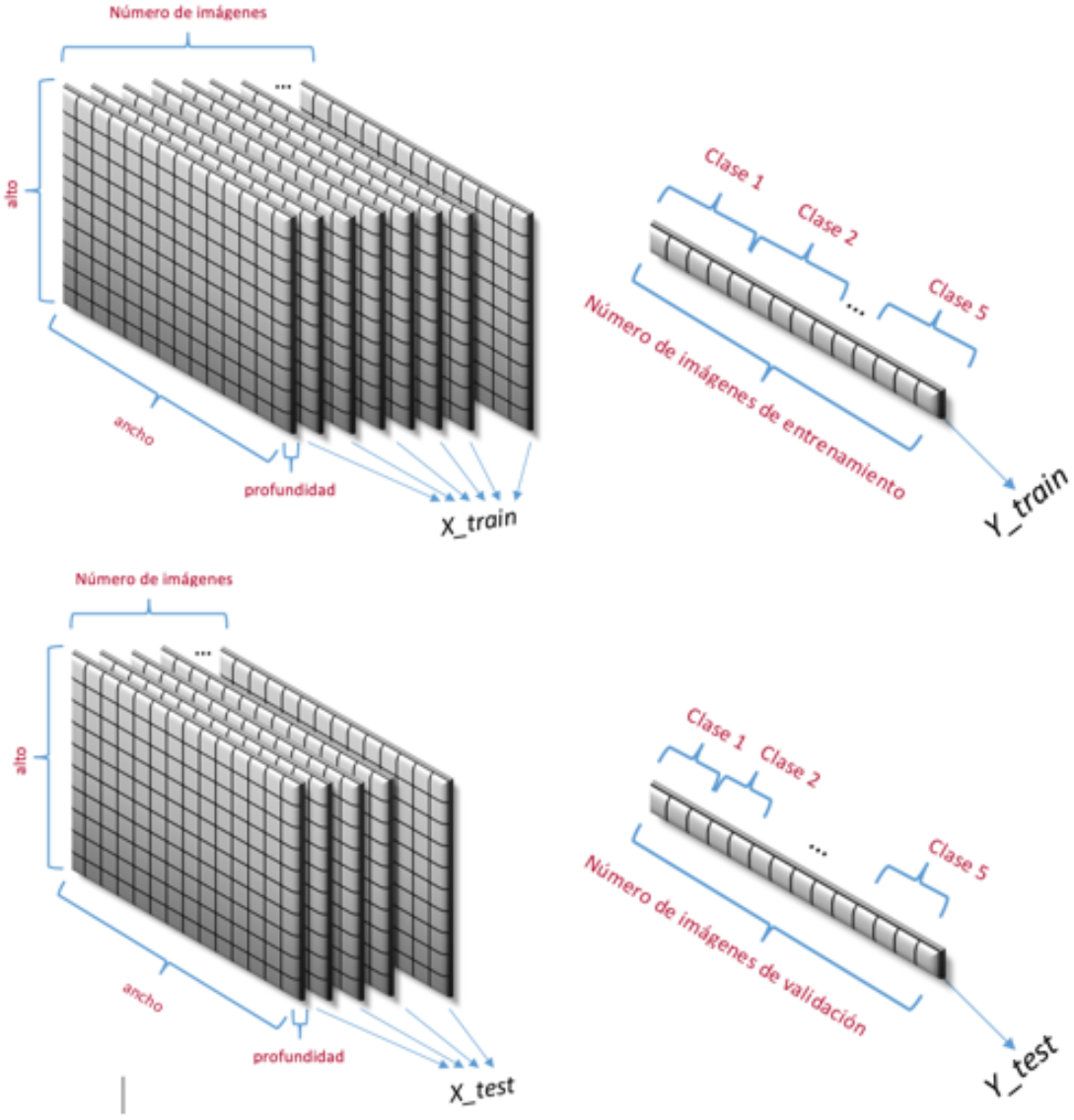

Figura 4.7: Formato de matrices 4D y vectores de categorías.

Posteriormente, se vinculó la creación de imágenes de datos aumentados (*image Data Augmenter*), que permitió generar un conjunto de datos (*AugX_train y AugX_test*) de las imágenes de entrenamiento y validación tales como rotación y traslación según el número de píxeles deseados. Seguidamente, se muestran los valores establecidos para estos parámetros:

- Rotación: [-20°...20°].
- Traslación (eje x y eje y): [-40...40].

- AugX_train: variable donde fueron almacenados *X_train* y *Y_train* respectivamente.

- AugX_test: variable donde fueron almacenados *X_test* y *Y_test* respectivamente.

Durante el cuarto paso, se especificaron las opciones de entrenamiento en las imágenes definiéndolas según el criterio del usuario, estas especificaciones estuvieron integradas por las siguientes opciones: **Tamaño de filtro, número de filtros** y **paso o Stride** [48], además del número de épocas, considerando que la variación de estas opciones alteraría el valor de exactitud resultante al final del algoritmo; fue necesario convertir el valor de entrada de estas especificaciones de texto a número por medio de variables tipo double. También tuvo que hacerse uso de la opción "Plots: trainig-progress" del toolbox de CNN Matlab para graficar el progreso de la validación. En este paso se determinaron los siguientes parámetros:

- Tamaño del filtro: [tf tf].

- Número de filtros: nf.

- Paso o Stride: [ps ps].

- Epochs: me.

En el siguiente paso se definió la arquitectura de la red neuronal convolucional, donde la capa fue establecida a partir del valor de las opciones anteriores además de otras como "Padding". Después de ésta se continuó con la definición de la función de activación también llamada "Relu" y con la operación de "Max Pooling" buscando eliminar información irrelevante durante el proceso. Finalmente, se definió la salida de la red en relación con el número de clases (5) por medio de la función "Fully conected layers". Se decidió usar una capa (sin utilizar las operaciones anteriormente mencionadas) porque de otra forma se hubiesen eliminado detalles importantes en la imagen.

Para realizar las pruebas, se tomaron en cuenta distintos tamaños de filtro, pudo evidenciarse a partir de ello que al designarse un tamaño de filtro de alto valor, se obtuvieron mejores resultados de exactitud; por ello, se decidió dar a los filtros tamaños cuadrados con distintas cantidades de píxeles con el fin de verificar cuál de ellos devolvía mayor porcentaje de validación (el tamaño cuadrado aparece como común denominador en los trabajos realizados con redes neuronales, sin embargo, se realizaron pruebas con tamaños de filtro rectangulares sin obtener resultados que llevaran a mejorar el proceso de aprendizaje). Además del tamaño, se realizaron pruebas variando el paso del filtro o Stride, siendo un paso igual a la mitad del valor de tamaño del filtro, la opción que mejores resultados arrojó en términos de tiempo y tendencia de las curvas de entrenamiento [49]. Finalmente, el número de filtros mayormente observado en la literatura fue de 25; sin embargo, aunque se realizaron pruebas con este valor, se decidió cambiar este parámetro en otras cantidades para observar la mejora en la exactitud del algoritmo (ver Tabla 4.1).

Número de imágenes	TF: 526 NF: 25 S: 263	TF: 526 NF: 32 S: 263	TF: 526 NF: 64 S: 263	TF: 300 NF: 25 S: 150	TF: 300 NF: 32 S: 150	TF: 300 NF: 64 S: 150	TF: 263 NF: 25 S: 132	TF: 263 NF: 32 S: 132	TF: 263 NF: 32 S: 132
10	**40**	**70**	**60**	**40**	**40**	**50**	**30**	**40**	**40**
15	**30**	**40**	**40**	**20**	**40**	**50**	**30**	**60**	**40**
20	**50**	**40**	**50**	**40**	**40**	**30**	**40**	**40**	**40**
25	**60**	**30**	**20**	**40**	**40**	**30**	**40**	**30**	**40**
30	**30**	**50**	**50**	**50**	**50**	**60**	**40**	**40**	**30**

Cuadro 4.1: Porcentaje de validación en diversas combinaciones.

A partir de lo anterior, se eligieron los parámetros con los que se obtuvo un porcentaje de exactitud mayor en uno de sus números de imágenes, estos parámetros fueron TF: 526, NF: 32, Stride: 263. Los valores para estos parámetros en cada número de imágenes fueron contenidos en una tabla (ver Tabla 5.2).

Después de identificar el mayor porcentaje de validación, se indagaron los parámetros establecidos para éste, buscando utilizar las mismas imágenes de esta prueba para las demás aumentando el número de imágenes (pruebas con otros números de imágenes); para ello fue necesaria la implementación de ciertas líneas de código que guardaron las imágenes recogidas en la prueba en una base de datos y su uso para pruebas posteriores, teniéndose así la necesidad de obtener aleatoriamente sólo la cantidad faltante (ver Tabla 5.3).

Se realizaron las matrices de confusión para los distintos números de imágenes; sin embargo, el número de imágenes escogido fue de diez pues entregó un mejor porcentaje de exactitud en el entrenamiento que los demás números. La matriz de confusión para la prueba de diez imágenes (ver Figura 5.5) permitió corroborar la exactitud de 70 %; éste y otros parámetros fueron contenidos en una tabla (ver Tabla 5.4).

Capítulo 5

RESULTADOS Y DISCUSIONES

5.1. SPEEDED UP ROBUST FEATURES

En esta técnica fue necesario utilizar la minería de datos para clasificar las imágenes entre las posibles clases tal y como se mencionó en el capítulo anterior. A partir de esto, se realizó la matriz de confusión que se expone a continuación:

Predicciones	Arma 1	502	178	125	89	137
	Arma 2	190	564	147	198	227
	Arma 3	229	262	870	181	207
	Arma 4	100	114	58	649	109
	Arma 5	127	112	30	113	548
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 5.1: Matriz de confusión para prueba SVM Coarse.

Para esta técnica, la matriz fue diseñada para la contabilización de los puntos de interés de las 50 vainillas pertenecientes a cada arma (la primera de ellas solo contó con 49 imágenes puesto que durante la toma de datos se tuvo un percance con una de las vainillas); cada imagen tuvo 82 puntos de interés, al multiplicarlos por la cantidad total de imágenes (49 en la primer clase y 50 en las demás) se obtuvieron valores de 1140 para la primer clase y 1230 para las demás. Esto quería decir que la suma total de las predicciones en cada clase debía dar los valores anteriormente mencionados. A pesar de haber usado dos clasificadores (KNN y SVM) y que cada uno de ellos brindó tres distintos subtipos de clasificación (distancias Euclidiana, CityBlock y Mahalanobis para KNN y kernels Coarse Gaussian, Medium Gaussian y Fine Gaussian para SVM), sólo se adjuntó en este capítulo la matriz de confusión que contuvo el mejor resultado de exactitud y por tanto realizó la mejor validación (ver Figura 5.1); este valor se obtuvo con el clasificador SVM y específicamente con el subtipo Coarse Gaussian. Posteriormente se adjuntó la tabla

con los valores de los parámetros de validación (ver Tabla 5.1).

Parámetro	Valor porcentual
Exactitud	51,53
Error	48,47
Sensibilidad	51,53
Especificidad	48,47

Cuadro 5.1: Parámetros de validación para pruebas SVM Coarse.

En el caso de la clasificación de armas, al ser más de dos variables y no poder contenerse en una decisión de sí o no, no pudieron calcularse con precisión la tasa de verdaderos negativos y falsos negativos; esto hizo que los porcentajes de exactitud y error coincidieran con los de sensibilidad y especificidad respectivamente.

Pudo evidenciarse, con lo anterior, que la técnica SURF no entregó un porcentaje de exactitud que generara confiabilidad para la clasificación correcta de nuevas imágenes. Este bajo nivel pudo deberse al ruido presente en ellas, el cual no permitió la detección de ciertos patrones repetitivos entre las vainillas de cada arma. Esto puede deberse a fallas durante la toma de imágenes entre las que se citan cambios en la iluminación y cambio del zoom en cada una; también, el fondo y el tamaño de ésta juegan un papel relevante a la hora de clasificar correctamente a la misma. La elección de los puntos clave tiene gran relevancia en esta técnica y, con los factores mencionados anteriormente, se pudieron generar nuevas regiones que desviaron los verdaderos puntos de interés para la clasificación.

5.2. CONVOLUTIONAL NEURAL NETWORKS

La gráfica 5.2 se obtuvo después del proceso de entrenamiento y validación de la red neuronal convolucional:

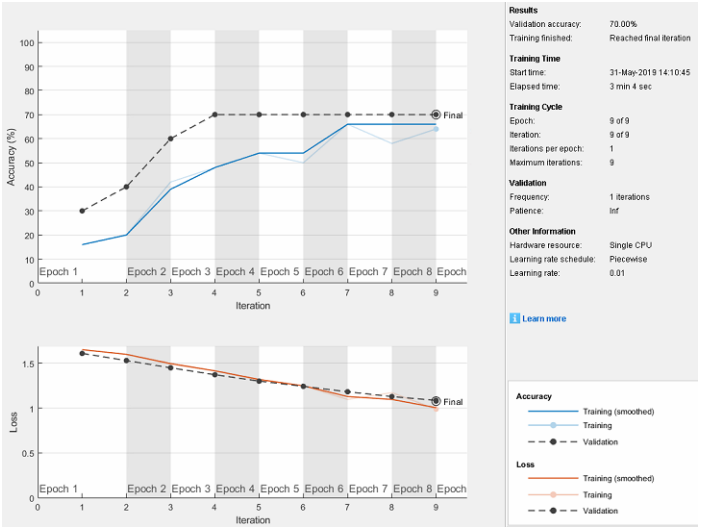

Figura 5.2: Resultados de entrenamiento y validación para 10 imágenes.

A partir de esto, pudo verse un comportamiento similar entre las señales de entrenamiento y validación; lo anterior es muy conveniente ya que para que una red neuronal se considere correcta, debe existir una relación de similitud alta entre ambas. pudo notarse que los mejores resultados fueron adquiridos con un número de diez imágenes y que sus parámetros fueron: Tamaño de filtro de 526 x 526, Stride de 263 x 263 y Número de filtros de 32. A partir de allí, se designaron estos mismos valores para los parámetros de las pruebas con otros números de imágenes (manteniendo los anteriores valores fijos y variándose solamente el número de imágenes, puesto que realizar cambios en distintas variables a la vez generó confusión a la hora de identificar aquellas que significaron alteraciones en el porcentaje de exactitud) obteniéndose lo siguiente:

Número de imágenes	**Porcentaje de exactitud**
10	70
15	50
20	60
25	50
30	70

Cuadro 5.2: Porcentajes de exactitud para distintos números de imágenes.

Por medio de la tabla anterior (ver Tabla 5.2), se realizó una gráfica (ver Figura 5.3) para observar los números de imágenes que tuvieron mayor eficacia en la clasificación de las armas.

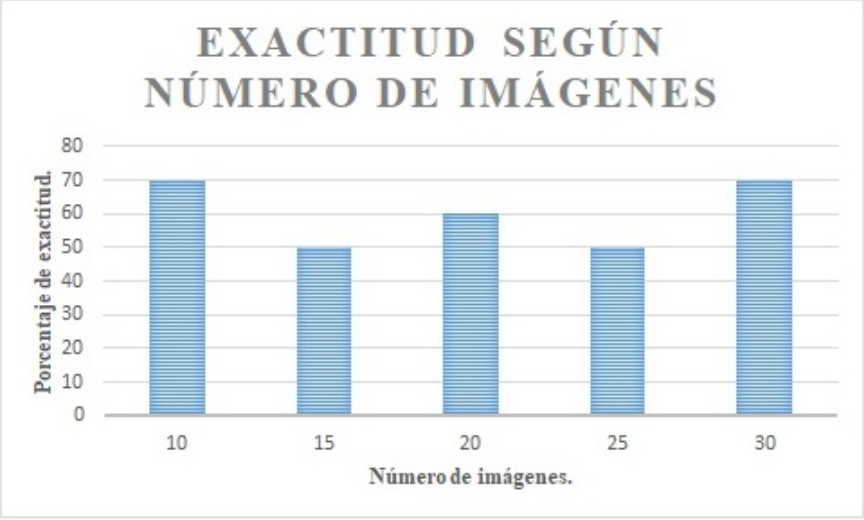

Figura 5.3: Exactitud para distintos números de imágenes.

Se constató que el mayor porcentaje obtenido fue 70 % para 10 imágenes por clase. Se decidió almacenarlas en una base de datos para requerir menos imágenes aleatorias en los siguientes números, el resultado anteriormente obtenido mejoró al 70 % en las demás combinaciones exceptuando la combinación para 15 imágenes, que dio como resultado 50 % (ver Tabla 5.3).

Número de imágenes	Porcentaje de exactitud
10	70
15	50
20	70
25	70
30	70

Cuadro 5.3: Porcentajes de exactitud para distintos números de imágenes con 10 imágenes fijas.

Con ello, se realizó una gráfica en la que pueden observarse mejor los porcentajes obtenidos y la relación entre sí (ver Figura 5.4)

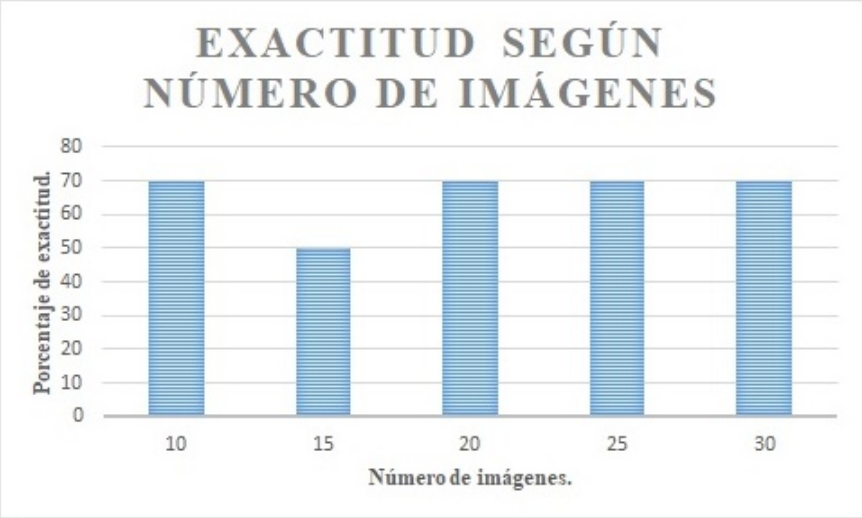

Figura 5.4: Exactitud para distintos números de imágenes con 10 imágenes fijas.

A partir de estos datos, se realizó la matriz de confusión para cada uno de los números de imágenes (ver Figura 5.5).

Predicciones	Arma 1	1,0	0,0	0,5	0,0	0,0
	Arma 2	0,0	1,0	0,0	0,0	0,0
	Arma 3	0,0	0,0	0,0	0,0	0,0
	Arma 4	0,0	0,0	0,0	1,0	0,5
	Arma 5	0,0	0,0	0,5	0,0	0,5
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 5.5: Matriz de confusión para validación de prueba con 10 imágenes.

Posteriormente se adjuntó la tabla con los valores de los parámetros de validación (ver Tabla 5.4).

Parámetro	Valor porcentual
Exactitud.	70
Error.	30
Sensibilidad.	70
Especificidad.	30

Cuadro 5.4: Parámetros de validación para pruebas con 10 imágenes.

En el caso de la clasificación de armas, al ser más de dos variables y no poder contenerse en una decisión de sí o no, no pudieron calcularse con precisión la tasa de verdaderos negativos y falsos negativos; esto hizo que los porcentajes de exactitud y error coincidieran con los de sensibilidad y especificidad respectivamente.

Este resultado permite mostrar que la técnica CNN es conveniente para el reconocimiento de armas de disparo idénticas aun cuando se vieron involucrados factores negativos durante el procesamiento de las imágenes; entre ellos se encontró la variación en el brillo y presencia de luz en cada imagen, esto puede ser explicado por las interrupciones que en ocasiones se tuvieron durante la toma de imágenes con el estereoscopio y por la variación de las constantes lumínicas entre un día y otro. Además de este factor, también se sumó el hecho de trabajar con vainillas percutidas por armas relativamente nuevas que aún no habían desarrollado su totalidad de características individuales para brindar un mayor reconocimiento en el arma (ya que las características de clase y subclase no aportaron información significativa en este caso por tratar de reconocerse armas de disparo idénticas). Teniendo en cuenta lo anterior, fue satisfactorio obtener una exactitud del 70 % para el algoritmo, puesto que en la búsqueda de trabajos realizados en este campo (reconocimiento de armas), aunque se encontraron resultados más altos tales como 88 % o 96 %, no se encontraron estudios con el objetivo de reconocer armas de disparo idénticas (como en este caso), siendo la similitud entre las mismas lo que más inconvenientes causa a la hora de reconocerlas.

Capítulo 6

TÉCNICA ALTERNATIVA: ANÁLISIS MORFOLÓGICO

6.1. PROCESAMIENTO

Buscando métodos que permitieran evidenciar si el diámetro del cráter de percusión presenta información relevante para la clasificación de las vainillas, se realizó un análisis morfológico [50] [51]. En esta técnica se determinaron el número de imágenes por clase y la cantidad de clases para, posteriormente, calcular el tamaño mínimo en ellas y definirlo para las demás. El procesamiento se realizó mediante las siguientes etapas:

- Carga de imágenes: Se determinó el número de imágenes, la clase que deseaba procesarse y el tamaño.
- Binarización: Se realizó un procesamiento morfológico (*imerode*) [52]. La apertura morfológica permitió una mejora de los resultados de la binarización y de erosión, ésta se hizo mediante la reconstrucción por dilatación. Los marcadores se eligieron de forma que eliminaran los detalles no deseados. Posteriormente, se prosiguió a la reconstrucción por dilatación; el efecto conseguido fue la preservación de las formas de los objetos que habían superado al marcado (ver Figura 6.1).

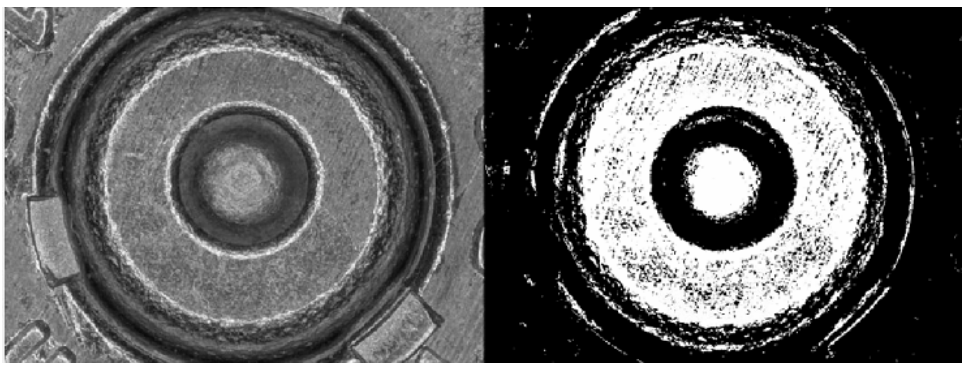

Figura 6.1: Contraste entre imagen en escala de grises e imagen procesada.

- Búsqueda del centro (1): Seguidamente, se comenzó la búsqueda del centro para la parte clara del cráter de percusión mediante dos cuadrados de longitudes 120x120 y 180x180 (las dimensiones de estos cuadrados fueron determinadas a prueba y error), calculando el número de píxeles para las áreas 1 y 2 (siendo estas la parte más clara y la más oscura respectivamente, tal como se muestra en la Figura 6.2).

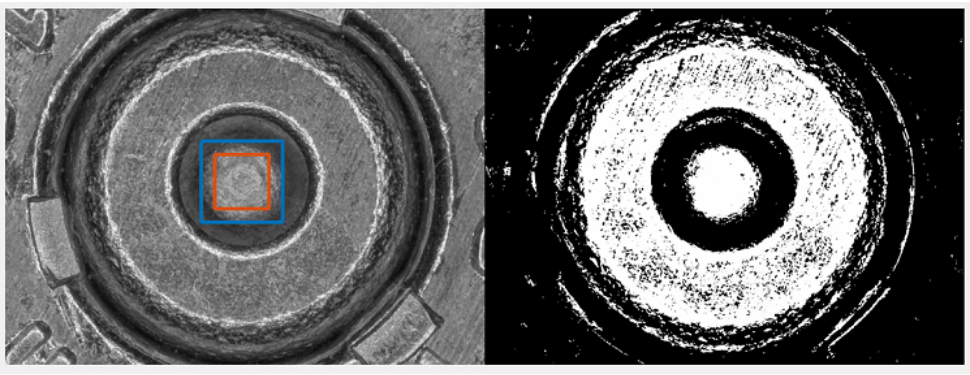

Figura 6.2: Contraste de búsqueda de centro en la parte más clara del cráter de percusión.

- Determinación del radio (1): Se buscó maximizar el área de interés, en este caso estuvo relacionada con el fondo del cráter de percusión (ver Figura 6.3).

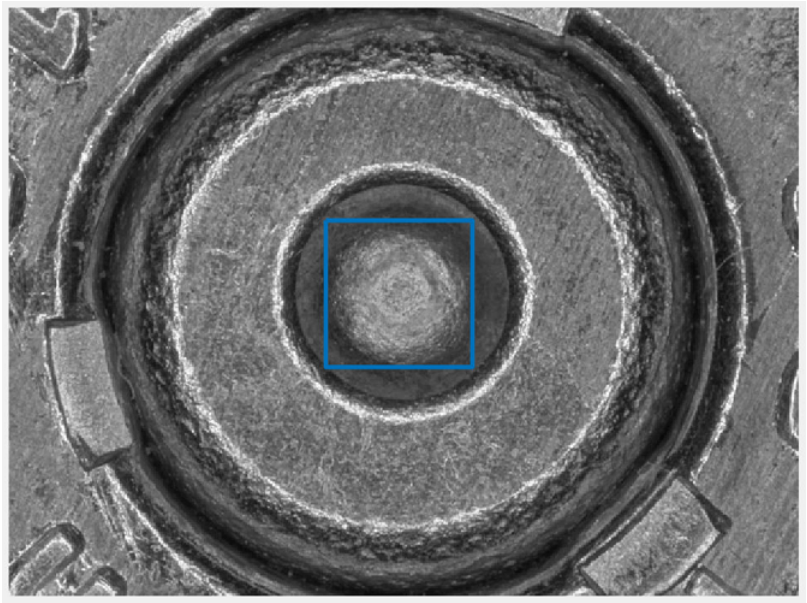

Figura 6.3: Determinación del radio para la zona más clara del cráter de percusión.

- Búsqueda del centro (2): La búsqueda del centro y medición del radio fueron realizadas de nuevo para la parte más oscura del cráter de percusión mediante dos cuadrados con longitudes 300x300 y 360x360, realizando además la eliminación de la zona localizada en la etapa anterior (zona más

clara) y la inversión de binarización; también se realizó la medición del radio para esta región. Con esto, se obtuvieron las Figuras 6.4 y 6.5.

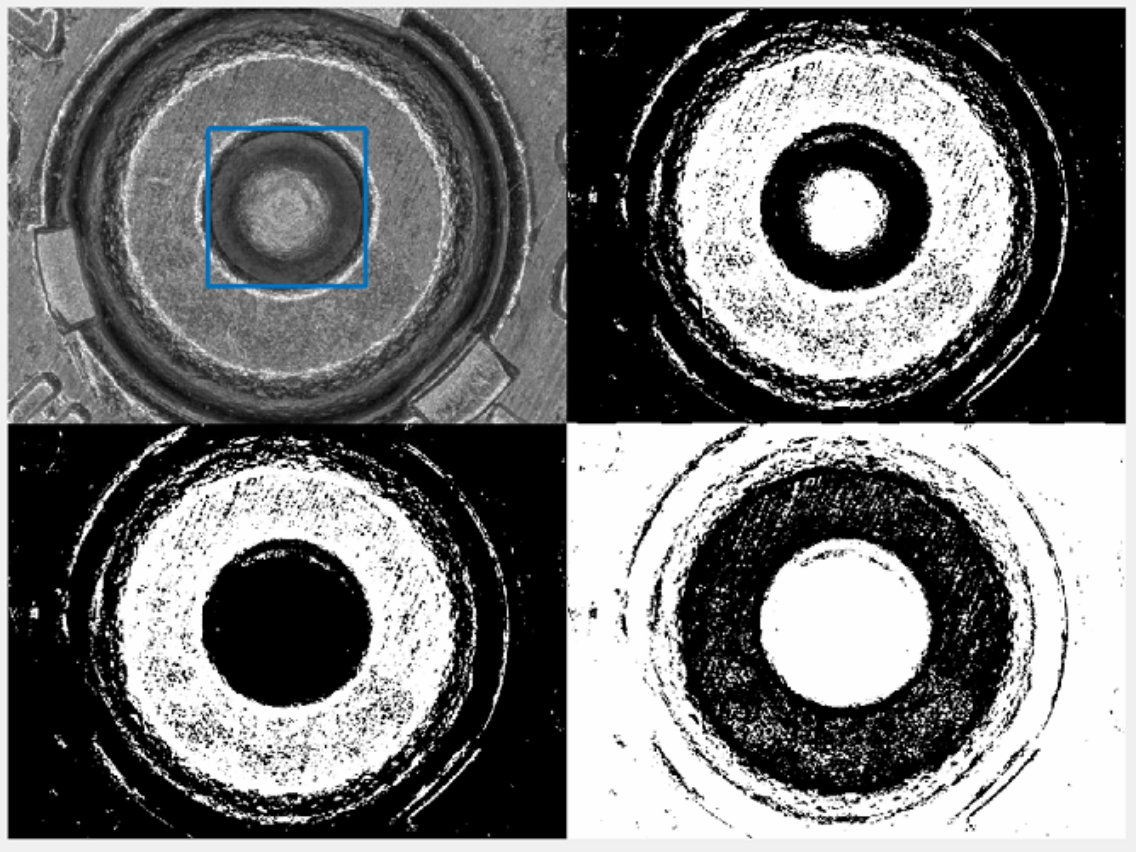

Figura 6.4: Búsqueda de centro para la zona oscura del cráter de percusión, eliminación de la zona más clara e inversión de negación.

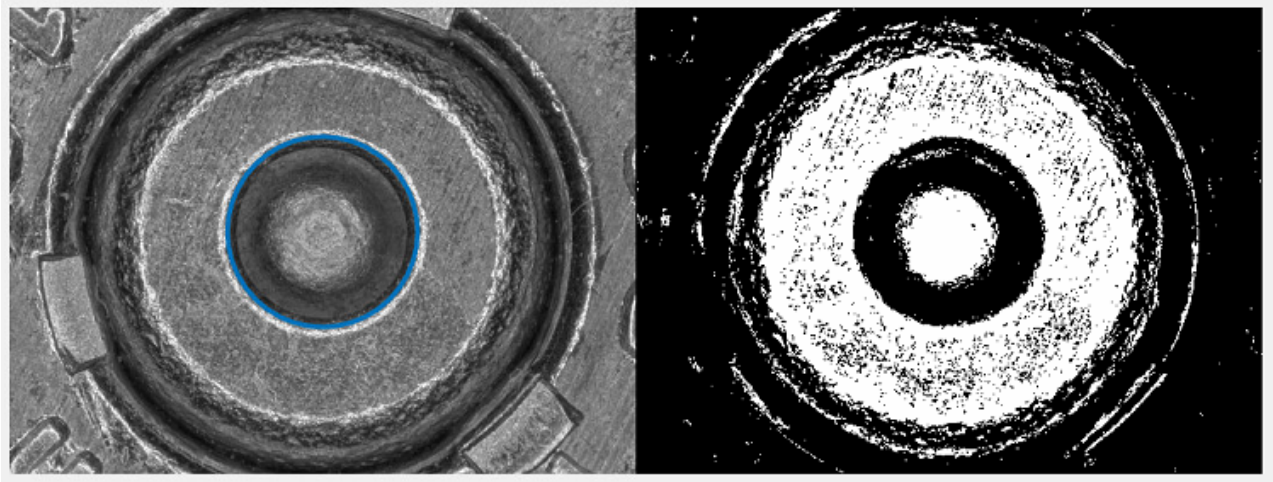

Figura 6.5: Medición de radio de la zona oscura en el cráter de percusión.

En la Figura 6.6 se presenta la extracción realizada al cráter de percusión.

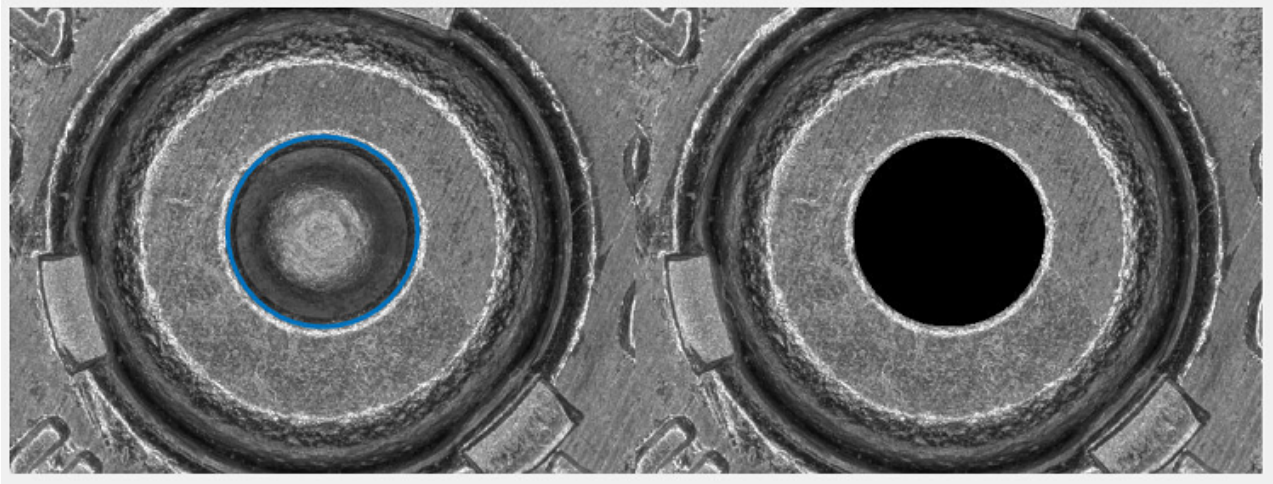

Figura 6.6: Extracción del cráter de percusión.

Posteriormente, se buscó el centro y radio para el fulminante en cada una de las imágenes y se obtuvo la Figura 6.7.

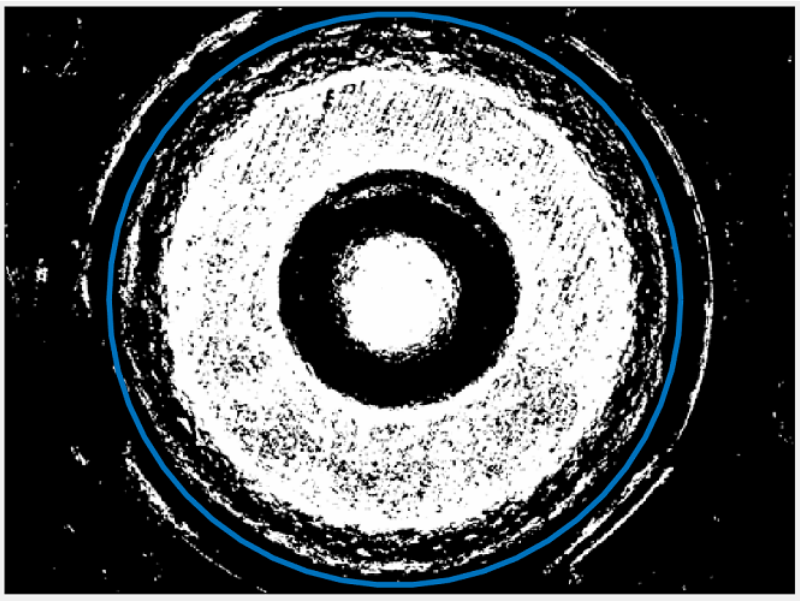

Figura 6.7: Búsqueda del centro para el fulminante de la vainilla.

Después de lo anterior, se buscó eliminar el fondo de las imágenes para impedir que éste desviara a la técnica de las características significativas en el fulminante (ver Figura 6.8). Sin embargo, no sólo era necesaria la eliminación del fondo sino que esto debía adicionarse a la extracción del cráter de percusión realizada en la etapa anterior; por lo anterior, se buscó dejar en las imágenes sólo la presencia del fulminante de cada una de las vainillas (ver Figura 6.9).

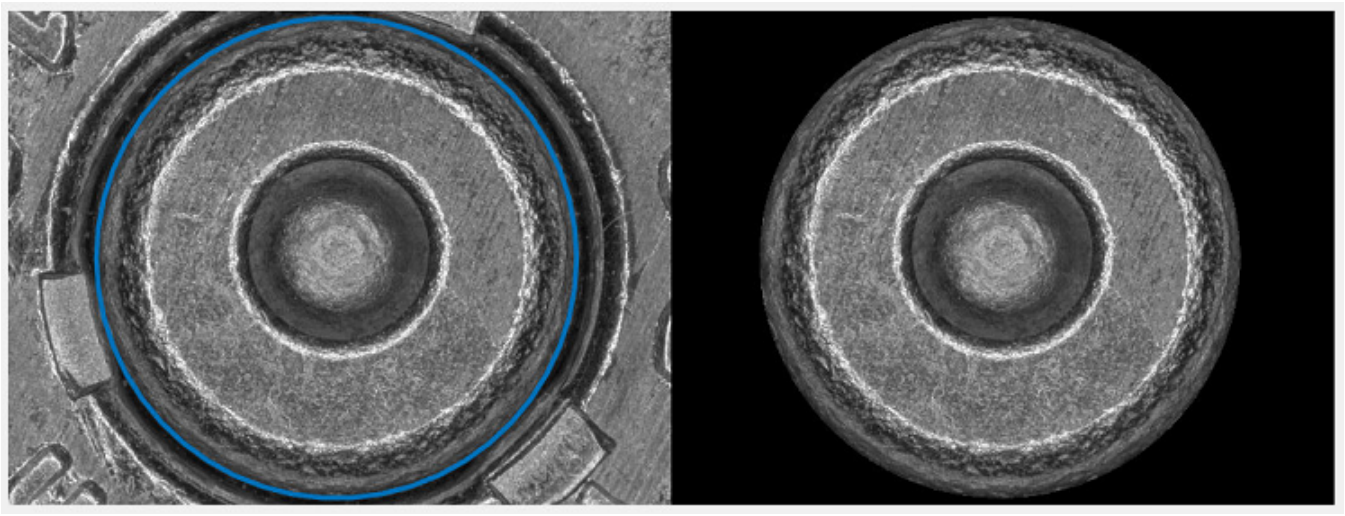

Figura 6.8: Eliminación de fondo en la imagen.

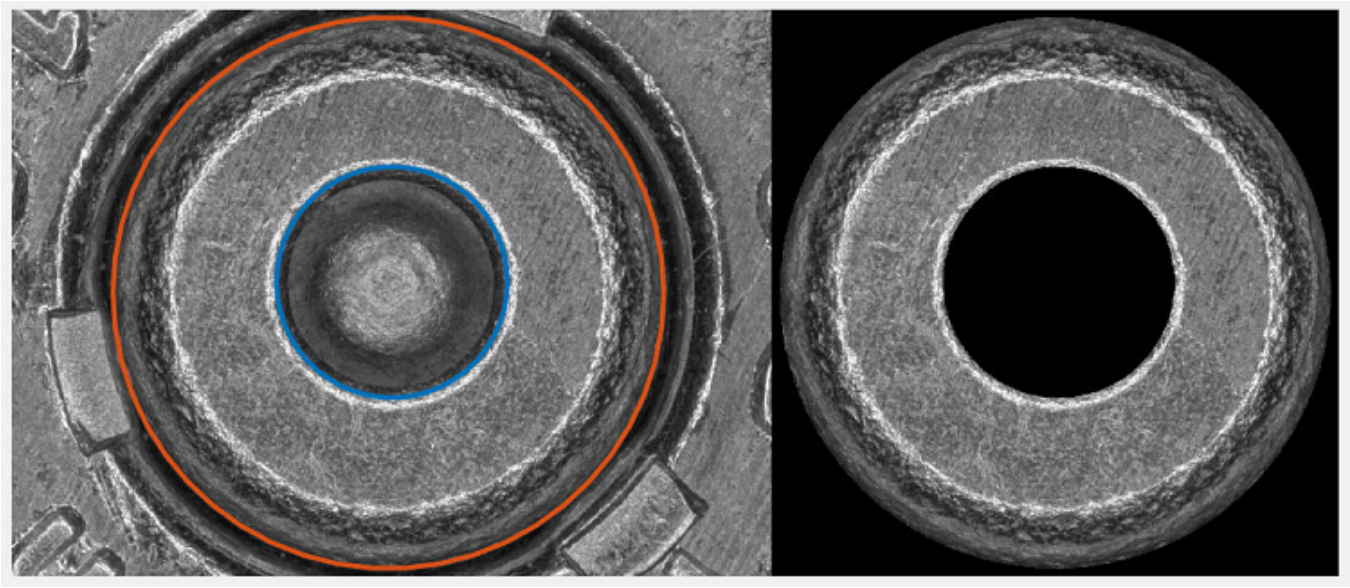

Figura 6.9: Extracción del fondo y del cráter de percusión en la vainilla.

Para la clasificación de las imágenes mediante la información anterior, se revisó que los diámetros encontrados en las imágenes coincidieran con la ubicación del cráter de percusión (los diámetros de las imágenes que no coincidan hicieron que éstas fueran descartadas en la etapa de clasificación); cabe destacar que algunos de los diámetros para cada clase no pudieron ser detectados por la técnica (no se detectaron sus bordes y se perdían características) y por ello también fueron descartados. Después de guardar los valores para los diámetros válidos por clase, se calculó su promedio y seguidamente su desviación estándar, graficando estos valores (ver Figura 6.10).

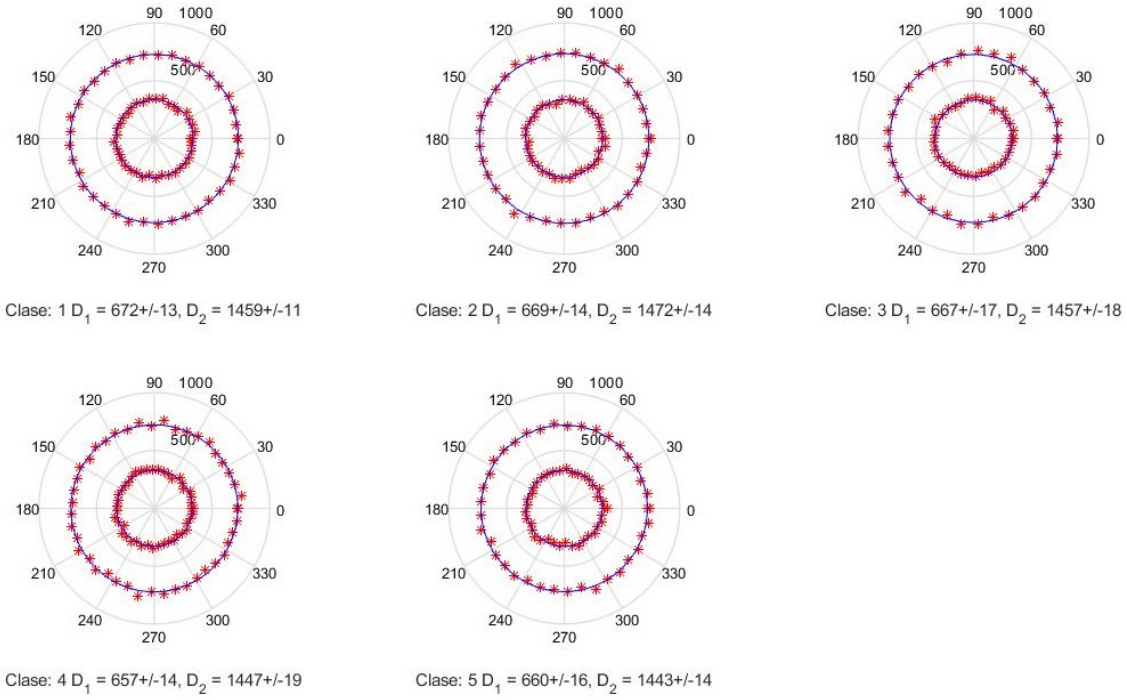

Figura 6.10: Ubicación de los diámetros del cráter de percusión y el fulminante alrededor del promedio.

Si los valores de desviación estándar para cada clase se traslapaban, esto significaba que no era posible clasificar a partir de esta técnica por la confusión con valores que podían pertenecer a varias clases. Para su mejor clasificación se realizó una campana de Gauss para los radios del cráter de percusión y fulminante (ver Figuras 6.11 y 6.12).

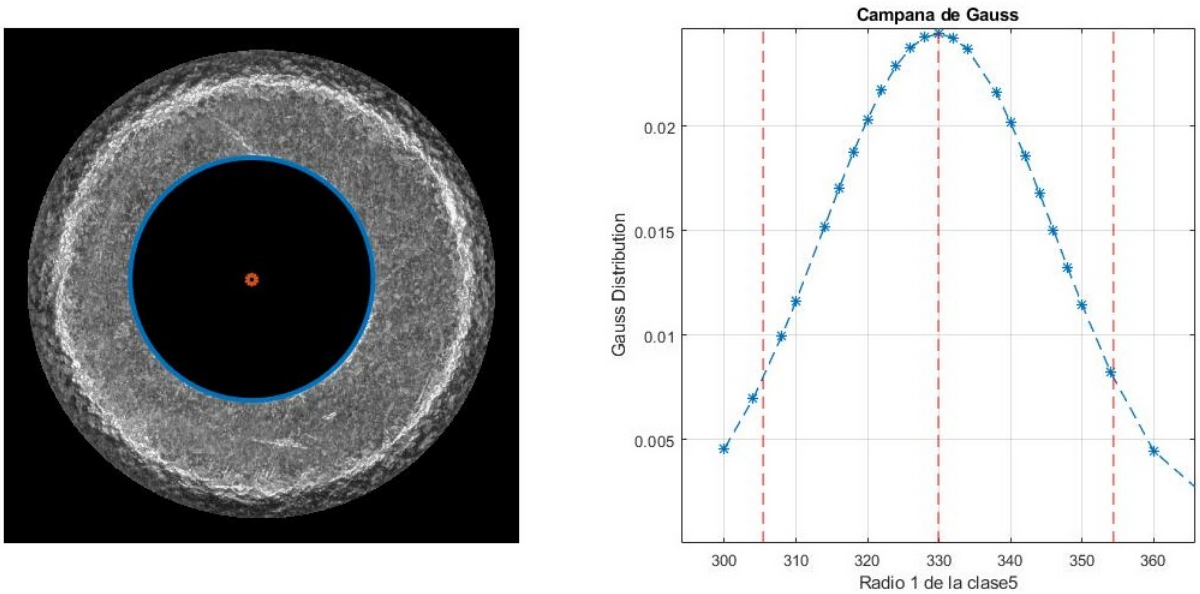

Figura 6.11: Campana de Gauss para radio del cráter de percusión en la clase 5.

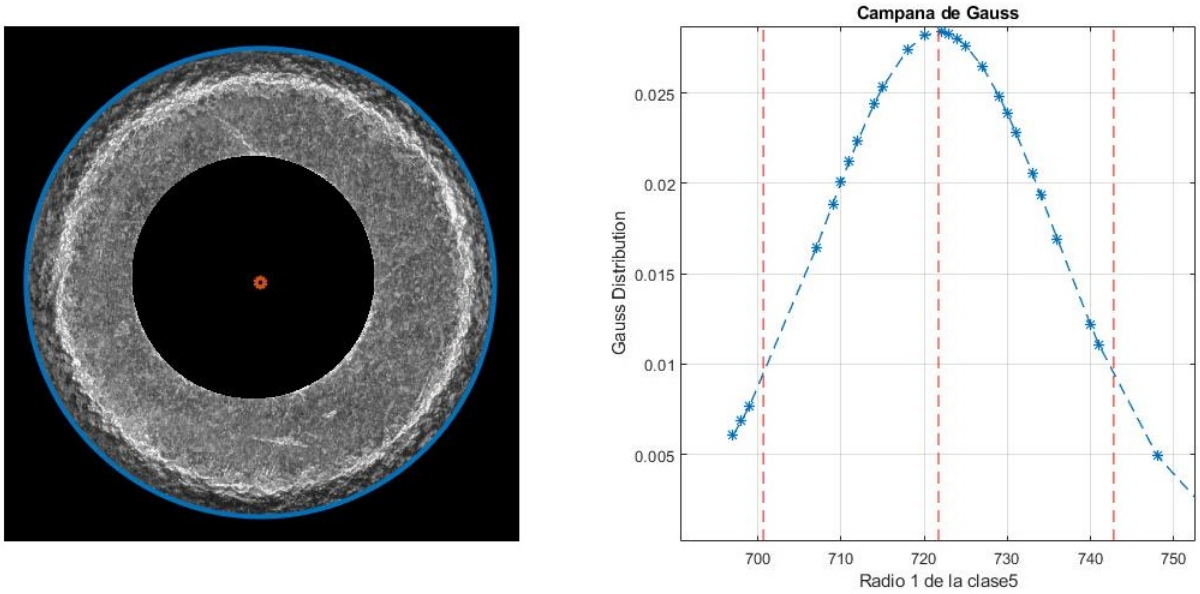

Figura 6.12: Campana de Gauss para radio del fulminante en la clase 5.

6.2. ANÁLISIS Y DISCUSIÓN DE RESULTADOS

A partir del promedio y la desviación estándar de los diámetros para cada clase, se realizaron las gráficas expuestas en el capítulo de Anexos (de la Figura 8.17 a la 8.28).

En base a las gráficas, se evidenció que los valores de diámetro entre vainillas no son muy diferenciables entre sí; es decir, los valores para una vainilla podían coincidir con los de otra de distinta clase, lo que hizo difícil la tarea de categorizarlas. Por esta razón, pudo notarse que esta técnica no es viable para la identificación y reconocimiento de armas idénticas y que incluso el microrrayado (característica clave del fulminante) no entrega información clave para facilitar este proceso y entregar resultados exactos.

Capítulo 7

CONCLUSIONES Y TRABAJOS FUTUROS

Durante el trabajo realizado, su obtuvieron ciertos inconvenientes para la aplicación de las técnicas en las imágenes conformantes de la base de datos ya que el ruido fue un elemento presente en muchas de ellas, por lo que el resultado de exactitud de las mismas se vio afectado en gran medida. Sin embargo, con las características que pudieron identificarse con cada una de las técnicas, se evidenció que las redes neuronales convolucionasles (CNN) brindan una mejor detección de los puntos de interés y por tanto entrenan y validan los datos de las imágenes con mayor exactitud que la técnica Speeded Up Robust Features (SURF). Además, aunque el costo computacional de la primer técnica en relación a la segunda es mucho más alto para el tipo de imágenes que se manejaron (de gran tamaño y con un formato más pesado que las comunes), se optimiza tiempo al realizar por su cuenta la clasificación de éstas de manera eficaz; esto le brinda una ventaja de peso a la hora de compararla con SURF aun sin tener en cuenta los porcentajes de exactitud obtenidos.

Para el presente proyecto, se estudiaron las características del fulminante; sin embargo, finalizando el proceso se llegó a la hipótesis de que los porcentajes de exactitud podrían mejorar si se evaluara la huella del eyector en cada imagen; lo anterior no pudo comprobarse por falta de tiempo, pero se plantea como una propuesta para trabajos futuros en esta área. La implementación de la técnica CNN, para detectar esta característica particular en las vainillas, podría hacer que mejore su exactitud (siendo ésta alta teniendo en cuenta los problemas de ruido presentes en las imágenes); es recomendable que para su correcta aplicación, en la etapa de recolección de imágenes y creación de la base de datos, se tenga especial cuidado con los factores externos, es decir, que no hayan episodios de iluminación repentina en la captación de estas imágenes para que no traduzcan ruido en la etapa de procesamiento de las mismas.

El resultado con la técnica SURF no fue el esperado y se notó poco rendimiento para el objetivo de clasificar las vainillas provenientes de armas de disparo idénticas; sin embargo, la técnica CNN arrojó unos resultados mejores frente a los cuales se puede esperar un incremento al realizar un proceso más detallado en la primera y segunda etapa del proyecto (captación de imágenes y procesamiento de las

mismas). Cabe destacar que este fue el primer acercamiento que se ha realizado con esta técnica al área balística, específicamente al reconocimiento de armas de disparo idénticas, puesto que, aunque se ha vinculado en la identificación de armas a las redes neuronales, no se ha hecho este proceso para armas idénticas y tampoco se ha explorado concretamente la técnica de redes convolucionales; así mismo, la base de datos en los artículos estudiados fue mucho mayor a la procesada durante el proyecto. Por lo anterior, los resultados obtenidos fueron satisfactorios y permitieron concluir que esta técnica puede ser clave en el área de balística con estudios futuros, ya que la incursión en la neurocomputación para la identificación de armas sigue avanzando y podría presentar grandes ventajas en este tema.

Capítulo 8

ANEXOS

A continuación, se presentan los anexos para el presente trabajo de investigación formativa en el siguiente orden:

Pre-procesamiento necesario para la aplicación de la técnica SURF

- Lectura de la imagen

- Conversión a escala de grises

- Detección de bordes

- Segmentación de la imagen

Obtención de características mediante la técnica SURF

Aplicación de la redes neuronales convolucionales CNN

- Determinación del número de imágenes y su respectivo tamaño

- Carga de la imágenes en una base de datos

- Creación de las imágenes de datos aumentados

- Especificación de las opciones de entrenamiento

- Determinación de la arquitectura de la red

- Inicio del entrenamiento

Análisis morfológico

- Carga de imagen

- Binarización

- Búsqueda del centro y determinación del radio del cráter de percusión

- Búsqueda del centro y determinación del radio del fulminante

- Extracción de las características del cráter de percusión y del fulminante

8.1. ANEXO A: Código de pre-procesamiento

```
I=imread('B.1_1_38.3X_C0.tif');

    J = rgb2gray(imresize(I, 0.5));

     figure(1)
     imshow(J)

     THRESH=graythresh(J);
     SIGMA=4.0

B=edge(J,'canny',THRESH,SIGMA);

figure(2)
imshow(B)
Rmin=135
Rmax=240
[centersBright, radiiBright] = imfindcircles(B,[Rmin
Rmax],'sensitivity',0.97);

 % Plot dark circles in dashed red boundaries

figure (3)
imshow(J)
viscircles(centersBright, radiiBright,'LineStyle','-');

figure(4)

x0=centersBright(1)-radiiBright-10;
y0=centersBright(2)-radiiBright-10;
w=2*radiiBright+20;
h=2*radiiBright+20;

K=imcrop(J,[x0 y0 w h]);
imshow(K)
```

8.2. ANEXO B: Código SURF

```
points = detectSURFFeatures(K);
% strongest = points.selectStrongest(93);
strongestPoints = selectStrongest(points,10)
[features, valid_points] = extractFeatures(K, strongestPoints);
 figure;
 imshow(K);
hold on;
plot(strongestPoints);
% surfPoints/plot(axesHandle,Name,Value) additional control for
the plot method specified by one or more Name,Value pair
arguments.
%-----'ShowScale' — Display proportional circle around feature
true (default) | false
%      Display proportional circle around feature, specified as
a comma-separated pair consisting of 'ShowScale' and the logical
true or false.
%      When you set this value to true, the object draws a
circle proportional to the scale of the detected feature.
%      The circle contains the feature point located at its
center. When you set this value to false, the object turns the
display of the circle off.
%      The algorithm represents the scale of the feature with a
circle of 6*Scale radius.
%      The SURF algorithm uses this equivalent size of circular
area to compute the orientation of the feature
%-----'ShowOrientation' — Display a line corresponding to
feature point orientation false (default) | true
%      Display a line corresponding to feature point
orientation, specified as a comma-separated pair consisting of
'ShowOrientation'
%      and the logical true or false.When you set this value to
true, the object draws a line corresponding to
%      the point's orientation. The object draws the line from
the feature point location to the edge of the circle, indicating
the scale.
% [x]=valid_points.selectStrongest(30);
%  plot(valid_points.selectStrongest(100),'showScale',true);
 %localizacion de los puntos claves en 'x' y 'y'
 [y] = valid_points.Location;
```

8.3. ANEXO C: Código CNN

```
%% Paso 1
% Número de imágenes por cada clase (m) para el entrenamiento
% El número total de imégenes de entrenamiento (N)
% Un parámetro adicional para el entrenamiento es el número de épocas (me: MaxEpochs)
clear all; close all;
m           = input('Ingrese el número de imágenes     (m): ')+2;   % Número de imágenes por
clase
clc
nClases     = 5;                                                    % Número de clases
N           = m*nClases;                                            % Número de total de
imagenes de entrenamiento
% 1.1 Cálculo del tamaño de la imagen
% Es importante tener en cuenta que en Matlab la carpeta de trabajo (Current Folder) debe
contener las imágenes en carpetas con el nombre de su respectiva clase.
imgFolder   = fullfile('Clase_1');
imgSets     = imageSet(imgFolder,'recursive');
tam         = size(imread(imgSets.ImageLocation{2}));
disp('Paso 1. Determinar el número de imagenes y el tamaño')
disp(['* Tamaño de la imagen:     ',num2str(tam)])
disp(['* Imágenes por cada clase: ', num2str(m)])
disp(['* Clases:                  ', num2str(nClases)])
disp(['* Total imágenes:          ', num2str(N)])
disp('-----------------------------------------------')
```

```
% Paso 2. Cargar imágenes en una base de datos adicional
[file,path] = uigetfile('*.mat');
load(file)
disp(['Base de datos: ',file,' cargada'])
clear file path
X_train     = zeros(tam(1),tam(2),1,N);
n = 1; %numImagClase = [49 50 50 50 50];

%i_1 = zeros(m,5);
alea2
disp(i_2(1:end,:))
for k=1:nClases
    switch k
    case 1
        imgFolder = fullfile('Clase_1');
        imgSets = imageSet(imgFolder,'recursive');
    case 2
        imgFolder = fullfile('Clase_2');
        imgSets = imageSet(imgFolder,'recursive');
    case 3
        imgFolder = fullfile('Clase_3');
        imgSets = imageSet(imgFolder,'recursive');
    case 4
        imgFolder = fullfile('Clase_4');
        imgSets = imageSet(imgFolder,'recursive');
    otherwise
        imgFolder = fullfile('Clase_5');
        imgSets = imageSet(imgFolder,'recursive');
    end

    for i = 1:m
        X_train(:,:,:,n) = im2double(imread(imgSets.ImageLocation{i_2(i,k)}));
        n = n+1;
    end
end
Y_train = categorical([0*ones(m,1);ones(m,1);2*ones(m,1);3*ones(m,1);4*ones(m,1)]);
ml = size(i_1,1);
idx=[];
for i = 1:nClases
    idx =  [idx;(i-1)*m+(ml-1)];
    idx = [idx;(i-1)*m+ml];
end
```

```
% Se crea la matriz de imágenes de prueba usando idx
X_test = X_train(:,:,:,idx);
% Se crea el vector de categorias correspondiente a la matriz de imágenes
% de prueba usando idx
Y_test = Y_train(idx);

% Se borran las imágenes de prueba de la matriz de imágenes de
% entrenamiento
X_train(:,:,:,idx) = [];
% Se borran las categorías de prueba del vector de validación de las
% imágenes de entrenamiento
Y_train(idx) = [];

% se calculan los tamaños
sX_train = size(X_train);sY_train = size(Y_train);
sX_test = size(X_test);  sY_test = size(Y_test);

% save('baseDatos15img.mat', '-v7.3') ;
% Se muestran los resultados

disp('Paso 2. Cargar imágenes en una base de datos')
disp(['Entrenamiento (X_train): ',num2str((sX_train(1))),'x',...
                                  num2str((sX_train(2))),'x',...
                                  num2str((sX_train(4)))])
disp(['Entrenamiento (Y_train): ',num2str((sY_train(1))),'x',...
                                  num2str((sY_train(2)))])
disp(['Prueba        (X_test) : ',num2str((sX_test(1))),'x',...
                                  num2str((sX_test(2))),'x',...
                                  num2str((sX_test(4)))])
disp(['Prueba        (Y_test) : ',num2str((sY_test(1))),'x',...
                                  num2str((sY_test(2)))])
%disp(i_1)
disp('-------------------------------------------------')

clear i idx imgFolder imgSets k n ans il numImagClase uno sX_train ...
      sY_train sX_test sY_test Num Numa M1 m1% nClases N m me



%% 3. Crear imágenes de datos aumentados imageDataAugmenter
% Un aumentador de datos imagen configura un conjunto de opciones para el aumento de la imagen,
tales como tamaño, rotación y reflexión de preprocesamiento.
angulo = 20;
trasla = 40;
imageAugmenter1 = imageDataAugmenter(...
    'FillValue',1,... 0(default)
    'RandRotation',[-angulo,angulo], ...
    'RandXScale',[1,1],...
    'RandYScale',[1,1],...
    'RandXTranslation',[-trasla trasla], ...
    'RandYTranslation',[-trasla trasla]);
imageAugmenter2 = imageDataAugmenter(...
    'FillValue',1,... 0(default)
    'RandRotation',[0,0], ...
    'RandXScale',[1,1],...
    'RandYScale',[1,1],...
    'RandXTranslation',[0 0], ...
    'RandYTranslation',[0 0]);
AugX_train = augmentedImageDatastore(tam,X_train,Y_train,'DataAugmentation',imageAugmenter1);
AugX_train.MiniBatchSize = 70;
AugX_test = augmentedImageDatastore(tam,X_test,Y_test,'DataAugmentation',imageAugmenter2);
AugX_test.MiniBatchSize = 32;
clear imageAugmenter1 imageAugmenter2
disp('Paso 3. Crear imágenes de datos aumentados')
disp(['Rotación           : [',num2str(-angulo),'...',...
                      num2str( angulo),']'])
disp(['Traslación (eje x) : [',num2str(-trasla),'...',...
                               num2str( trasla),']'])
disp(['Traslación (eje y) : [',num2str(-trasla),'...',...
                               num2str( trasla),']'])
disp('-------------------------------------------------')
```

```
%% 4. Especificar opciones de entrenamiento
%tf = inputdlg('Tamaño del filtro: ',...
%              'Opciones de entrenamiento', [1 50]);
op = inputdlg({'tf (Tamaño del filtro)',...
               'nf (Número de filtros' ,...
               'ps (Paso o Stride)    ' ,...
               'me (Epochs)'},...
               'Opciones de entrenamiento',...
               [1 30; 1 30; 1 30; 1 30]);
tf = str2double(op{1});%input('Tamaño del filtro: ');
nf = str2double(op{2});%input('Número de filtros: ');
ps = str2double(op{3});%input('Paso o el Stride : ');
me = str2double(op{4});%input('Ingrese el número de iteraciones (me): ');     % Número de
imágenes por clase
rng(1)
options = trainingOptions(...
    'sgdm',... 'sgdm' 'rmsprop' 'adam'
    'Plots','training-progress',... 'none'(default) 'training-progress'
    'Verbose',1,... 1(default) 0
    'VerboseFrequency',1,... 50(default)
    'MaxEpochs',me,... 30(default)
    'MiniBatchSize',70,... 128(default)
    'Shuffle','every-epoch',... 'once'(default) 'never' 'every-epoch'
    'ValidationData',AugX_test,...
    'ValidationFrequency',1,... 50(default)
    'ValidationPatience',Inf,... 5(default)
    'InitialLearnRate',0.01,... 0.01(default for stgd) 0.001(default for rmsprop and adam)
    'LearnRateSchedule','piecewise',... 'none'(default) 'piecewise'
    'LearnRateDropPeriod',100,... 10(default)
    'LearnRateDropFactor',0.01,... 0.1(default)
    'L2Regularization',0.01,... 0.0001(default)
    'Momentum',0.9,... 0.9(default)
    'CheckpointPath',''); % ''(default) 'C:\Users\Me\AppData\Local\Temp\'
%disp('-------------------------------------------------')
disp('Paso 4. Especificar opciones de entrenamiento')
disp(['* Tamaño del filtro:     [',num2str(tf),'x',num2str(tf),']'])
disp(['* Número de filtros:      ', num2str(nf)])
disp(['* Paso o Stride    :     [',num2str(ps),'x',num2str(ps),']'])
disp(['* Epochs           :      ', num2str(me)])
disp('-------------------------------------------------')
```

```
%% P aso 5. Definir la arquitectura de la red
% Actualización de datos:
% Tamaño filtro: tf
% Número de filtros: nf
% El número de pasos: ps
% Definir la arquitectura de la red neuronal convolucional.
layers = [
    imageInputLayer(tam) % 1579x2121x1

    % 1579x2121 = tamaño de la imagen, 526 =tamaño del kernel (ancho y alto),
25=número de kernel,

    convolution2dLayer([tf tf],nf,'Stride',[ps ps],'Padding',[0 0 0 0])
    batchNormalizationLayer
    reluLayer
%     maxPooling2dLayer(2,'Stride',2)
%
%      convolution2dLayer(8,16,'Stride',2)%,'Padding','same')
%      batchNormalizationLayer
%      reluLayer
%
%     maxPooling2dLayer(2,'Stride',2)
%
%     convolution2dLayer(3,32,'Padding','same')
%     batchNormalizationLayer
%     reluLayer
%     maxPooling2dLayer(2,'Stride',2)
    fullyConnectedLayer(5)
    softmaxLayer
    classificationLayer];
disp('Paso 5. Se define la arquitectura de la red')
disp(['* Tamaño del filtro:     [',num2str(tf),'x',num2str(tf),']'])
disp(['* Número de filtros:      ', num2str(nf)])
disp(['* Paso o Stride    :     [',num2str(ps),'x',num2str(ps),']'])
disp(['* Número de capas  :      ', num2str(l)])
disp('-----------------------------------------------')
%% Paso 6. Entrenamiento
net = trainNetwork(AugX_train,layers,options);
disp('Paso 6. Entrenar la red')
disp('Entrenamiento iniciado')
disp('-----------------------------------------------')
```

8.4. ANEXO D: Técnica alternativa (Análisis morfológico)

```
%% Etapa 1: Carga de imagen

clear all;clc
%nombre = 'Datos15Imag2.mat';
clase        = input('Seleccione la clase (1...5): '    );
n            = input('Cantidad de imágenes a procesar: '); %numImagClase(clase); %
clases       = {'Clase_1','Clase_2','Clase_3','Clase_4','Clase_5'};
imgFolder    = fullfile(clases{clase});
imgSets      = imageSet(imgFolder,'recursive');

numImagClase = [49 50 50 50 50];
j1 = 1:numImagClase(clase);
if(clase==1);j1([1,5,9,11,20,22,30,41,42,45,47,48]) = [];end% 6 41 47n 11n
[5,6,9,20,22,30,41,42,48]
if(clase==2);j1([1,6,7,8,15,16,17,18,22,23,26,38,49,50]) = [];end
if(clase==3);j1([1,8,9,10,21,23,24,25,26,27,28,33,38,44,46,48,49]) = [];end
if(clase==4);j1([1,7,14,24,44,47,50]) = [];end
if(clase==5);j1([1,7,13,17,28,32,35,42,44,45,47,48]) = [];end %i            =
randperm(numImagClase(clase),n); Indices = i;
i            = j1(1:n); Indices = i;
tam          = size(imread(imgSets.ImageLocation{2}));
disp('Paso 1. Determinar el número de imagenes y el tamaño')
disp(['* Tamaño de la imagen:     ',num2str(tam)])
disp(['* Imágenes por cada clase: ', num2str(n)])
disp(['* Clases:                  ', num2str(clase)])
%disp(['* Total imágenes:          ', num2str(N)])
disp('------------------------------------------------')
%% Etapa 2: Binarización

imgBWElimBorde = zeros(tam(1),tam(2),1,n);%umbral = 0.3924;
se = strel('disk',1);
for k=1:n
    %se = strel('disk',1);
    imgEnt = imread(imgSets.ImageLocation{Indices(k)});
    umbral=graythresh(imgEnt);
    imgEroGrises =  imerode(imgEnt,se);
    imgBWMask=imbinarize(imgEnt, umbral);
    imgBWMarcador = imerode(imgBWMask,se);
    imgReconst=imreconstruct(imgBWMarcador,imgBWMask);
    imgBWElimBorde(:,:,1,k) = imclearborder(imgReconst);%imgBWMask;%
    figure(k)
    imshow(imresize([imgEnt,255*imgBWElimBorde(:,:,1,k)],0.2))
    title('Imagen en escala de grises ... imagen procesada')
end
%imshow([imresize(imgEnt,0.25),imresize(imgBWElimBorde(:,:,1,k)*255,0.25)])
disp('Paso 2. Binarización')
disp(['* Tamaño de la imagen:     ',num2str(tam)])
disp(['* Imágenes por cada clase: ', num2str(n)])
disp(['* Clases:                  ', num2str(clase)])
%disp(['* Total imágenes:          ', num2str(N)])
disp('------------------------------------------------')
```

```
%% Etapa 3: Centro de la imagen
xc = round(tam(2)/2); yc = round(tam(1)/2); L = 120; L1 = 180;
% Busqueda del centro
b = 100000; co = zeros(n,2);
for i=1:n
    b = 100000; c = 0;
    I = imgBWElimBorde(:,:,1,i);
    for k=-150:5:150
        for j=-150:5:150
                A = sum(sum(I(yc-L +k:yc+L +k,xc-L +j:xc+L +j)));
                a = sum(sum(I(yc-L1+k:yc+L1+k,xc-L1+j:xc+L1+j)))-...
                    A;%sum(sum(I(yc-L +k:yc+L +k,xc-L +j:xc+L +j)));
            if((a/A)<b)
                b=a/A;
                co(i,:) = [yc+k xc+j];
            end
        end
    end
    imshow(imresize([imgEnt,255*imgBWElimBorde(:,:,1,i)],0.2)); hold on;
    plot(0.2*co(i,2),0.2*co(i,1),'*','linewidth',5); hold off
    title('Imagen en escala de grises ... imagen procesada (Centro)')
end
disp('Paso 3. Busqueda del centro 1')
disp(['* Centro de la imagen inicial  : ','Xc = ',num2str(xc)     ,' Yc =
',num2str(yc)])
disp(['* Centro de la imagen calculado: ','Xc = ',num2str(co(1,2)),' Yc =
',num2str(co(1,1))])
%disp(['* Total imágenes:          ', num2str(N)])
disp('-------------------------------------------------')
%% Etapa 4: Determinación del radio
r = zeros(1,n);
for j=1:n
    I = imgBWElimBorde(:,:,1,j);
for k=-100:100
    a = sum(sum(I(co(j,1)-L-k:co(j,1)+L+k,co(j,2)-L-k:co(j,2)+L+k)));
    b = numel(  I(co(j,1)-L-k:co(j,1)+L+k,co(j,2)-L-k:co(j,2)+L+k));
        if(a/b>=0.5)%0.65
            r(j) = k;
        end
end
end
disp('Paso 4. Calculo del radio 1')
disp(['* Radio de la imagen inicial  : ','r = ',num2str(L)])
disp(['* Radio de la imagen calculado: ','r = ',num2str(L+r(1))])
%disp(['* Total imágenes:          ', num2str(N)])
disp('-------------------------------------------------')
```

```
%% Etapa 5: Busqueda del centro 2
L = 300; L1 = 360;
b = 100000; co_1 = zeros(n,2); tol = 10;
for i=1:n
    b = 100000; c = 0;
    I = imgBWElimBorde(:,:,1,i); %figure(1); imshow(I);
    I(co(i,1)-120-r(i)-tol:co(i,1)+120+r(i)+tol,...
      co(i,2)-120-r(i)-tol:co(i,2)+120+r(i)+tol)=0; %figure(2); imshow(I);
    I = imcomplement(I); %figure(3); imshow(I);
    for k=-100:10:100
        for j=-100:10:100
                A = sum(sum(I(co(i,1)-L +k:co(i,1)+L +k,co(i,2)-L
+j:co(i,2)+L +j)));%sum(sum(I(yc-L1+k:yc+L1+k,xc-L1+j:xc+L1+j)));
                a = sum(sum(I(co(i,1)-L1+k:co(i,1)+L1+k,co(i,2)-L1
+j:co(i,2)+L1+j)))-...
                    A;%sum(sum(I(co(i,1)-L +k:co(i,1)+L +k,co(i,2)-L
+j:co(i,2)+L +j)));

            if((a/A)<=b)
                b=a/A;
                co_1(i,:) = [co(i,1)+k co(i,2)+j];
            end
        end
    end
end
disp('Paso 5. Busqueda del centro 2')
disp(['* Centro de la imagen inicial  : ','Xc = ',num2str(co(1,2)),' Yc =
',num2str(co(1,1))])
disp(['* Centro de la imagen calculado: ','Xc = ',num2str(co_1(1,2)),' Yc =
',num2str(co_1(1,1))])
disp('-------------------------------------------------')
```

```
%% Etapa 6: calculo del Radio 2
r_1 = zeros(1,n);
for j=1:n
    I = imgBWElimBorde(:,:,1,j);
    I(co(j,1)-120-r(j):co(j,1)+120+r(j),co(j,2)-120-r(j):co(j,2)+120
+r(j))=0;
    I = imcomplement(I);
for k=0:2:100
    a = sum(sum(I(co_1(j,1)-L-k:co_1(j,1)+L+k,co_1(j,2)-L-k:co_
1(j,2)+L+k)));
    b = numel(  I(co_1(j,1)-L-k:co_1(j,1)+L+k,co_1(j,2)-L-k:co_
1(j,2)+L+k));
    %A = sum(sum(I(co(j,1)-L -k:co(j,1)+L +k,co(j,2)-L -k:co(j,2)+L
+k)));%sum(sum(I(yc-L1+k:yc+L1+k,xc-L1+j:xc+L1+j)));
    %a = sum(sum(I(co(j,1)-L1-k:co(j,1)+L1+k,co(j,2)-L1-k:co(j,2)+L1+k)))-
A;%...
        %sum(sum(I(co(j,1)-L -k:co(j,1)+L +k,co(j,2)-L -k:co(j,2)+L +k)));
        if(a/b>=0.77)
        %if((a/A)<=b)
        %    b = a/A;
            r_1(j) = k;
        end
end
end
R_1 = L+r_1;
%%
% if(clase==1);R_1([5,9,11,20,22,30,41,42,45,47,48]) = [];end% 6 41 47n
11n [5,6,9,20,22,30,41,42,48]
% if(clase==2);R_1([1,6,7,8,15,16,17,18,22,23,26,38,49,50]) = [];end
% if(clase==3);R_1([8,9,10,21,23,24,25,26,27,28,33,38,44,46,48,49]) =
[];end
% if(clase==4);R_1([7,14,24,44,47,50]) = [];end
% if(clase==5);R_1([1,7,13,17,28,32,35,42,44,45,47,48]) = [];end
%%
disp('Paso 6. Calculo del radio 2')
disp(['* Radio de la imagen inicial  : ','r = ',num2str(L)])
disp(['* Radio de la imagen calculado: ','r = ',num2str(L+r_1(1))])
%disp(['* Total imágenes:          ', num2str(N)])
disp('-------------------------------------------------')
```

```
%% Busqueda del centro
% abajo
tol = 0.1; h = 0;
tf = [100 100]; st = [2 2]; centro = zeros(4,n); %n = 3;
for i=1:n
    u = 1;
    I = imgBWElimBorde(:,:,1,i);
    h = 0;
    for k=0:st(1):1500
        min1 = co_1(i,1)+300+r_1(i)      +k; if(min1<=1);min1=1;h=1;end
        max1 = co_1(i,1)+300+r_1(i)+tf(1)+k; if(max1>=1579);max1=1579;h=1;end
        min2 = co_1(i,2)-tf(2)/2           ;
        max2 = co_1(i,2)+tf(2)/2           ;
        A = sum(sum(I(min1:max1,min2:max2)))/10000;
        if(A<=tol || h==1)
            centro(u,i) = max1-tf(1)/2;
            u=u+1;
            break;
        end



%             cu1 = [min2 min1;max2 min1;...
%                    max2 max1;min2 max1;...
%                    min2 min1];
%             cu2 = [co_1(i,2)-(300+r_1(i)) co_1(i,1)-(300+r_1(i));...
%                    co_1(i,2)+(300+r_1(i)) co_1(i,1)-(300+r_1(i));...
%                    co_1(i,2)+(300+r_1(i)) co_1(i,1)+(300+r_1(i));...
%                    co_1(i,2)-(300+r_1(i)) co_1(i,1)+(300+r_1(i));...
%                    co_1(i,2)-(300+r_1(i)) co_1(i,1)-(300+r_1(i))];
%             imshow(I);hold on;
%             plot(cu1(:,1),cu1(:,2),'linewidth',3);
%             plot(cu2(:,1),cu2(:,2),'linewidth',3);
%             plot(co_1(i,2),co_1(i,1),'*','linewidth',3); hold off
%             pause(0.0000000001)



    end
    h = 0;
```

```
 % arriba
    for k=0:-st(1):-1500
        max1 = co_1(i,1)-300-r_1(i)      +k; if(max1>=1579);max1=1579;h=1;end%
if(min1<1   );min1=1   ;end
        min1 = co_1(i,1)-300-r_1(i)-tf(1)+k; if(min1<=1);min1=1;h=1;end%
if(max1>1579);max1=1579;end
        min2 = co_1(i,2)-tf(2)/2           ; %if(min2<1   );min2=1   ;end
j-tf(2)/2;%
        max2 = co_1(i,2)+tf(2)/2           ; %if(max2>2121);max2=2121;end
j+tf(2)/2;%
        A = sum(sum(I(min1:max1,min2:max2)))/10000;
        if(A<=tol || h==1)
            centro(u,i) = min1+tf(1)/2;%[k,j];
            u=u+1;
            break;
        end
    end
    h = 0;
% derecha
    for j=0:st(2):1500
        min1 = co_1(i,1)-tf(1)/2           ; %if(min1<1   );min1=1   ;end
co_1(i,1)+300+r_1(i)      +k;
        max1 = co_1(i,1)+tf(1)/2           ; %if(max1>1579);max1=1579;end
co_1(i,1)+300+r_1(i)+tf(1)+k;
        min2 = co_1(i,2)+300+r_1(i)      +j; %if(min2<1   );min2=1   ;end
j-tf(2)/2;%
        max2 = co_1(i,2)+300+r_1(i)+tf(2)+j; %if(max2>2121);max2=2121;end
j+tf(2)/2;%
        A = sum(sum(I(min1:max1,min2:max2)))/10000;
        if(A<=tol || h==1)
           centro(u,i) = max2-tf(2)/2;%[k,j];
           u=u+1;
           break;
        end
    end
    h = 0;
 % izquierda

    for j=0:-st(1):-1500
       max1 = co_1(i,1)+tf(1)/2           ; if(max1>=1579);max1=1579;h=1;end%
if(min1<1   );min1=1   ;end      co_1(i,1)-300-r_1(i)      +k;
       min1 = co_1(i,1)-tf(1)/2           ; if(min1<=1);min1=1;h=1;end%
if(max1>1579);max1=1579;end      co_1(i,1)-300-r_1(i)-tf(1)+k;
       max2 = co_1(i,2)-300-r_1(i)      +j; if(max2>=2121);max2=2121;h=1;end%
if(min2<1   );min2=1   ;end      j-tf(2)/2;%
       min2 = co_1(i,2)-300-r_1(i)-tf(2)+j; if(min2<=1);min2=1;h=1;end%
if(max2>2121);max2=2121;end      j+tf(2)/2;%
       A = sum(sum(I(min1:max1,min2:max2)))/10000;
       if(A<=tol || h==1)
           centro(u,i) = min2+tf(2)/2;%[k,j];
           u=u+1;
           break;
       end
    end
end
```

8.5. LISTA DE TABLAS

Nombre de la imagen	Parte de la vainilla	Iluminación circular
A.K#_V#_Z#_C0	Base	Completa
B.K#_V#_Z#_C0	Fulminante	Completa
C.K#_V#_Z#_C12-3	Fulminante	Superior y lateral derecha
D.K#_V#_Z#_C3-6	Fulminante	Lateral derecha e inferior
E.K#_V#_Z#_C6-9	Fulminante	Inferior y lateral izquierda
F.K#_V#_Z#_C9-12	Fulminante	Lateral izquierda y superior
G.K#_V#_Z#_C3-9	Fulminante	Lateral derecha e izquierda
H.K#_V#_Z#_C12-6	Fulminante	Superior e inferior
I.K#_V#_Z#_C3-9	Eyector	Lateral derecha e izquierda
J.K#_V#_Z#_C12-6	Eyector	Superior e inferior

Cuadro 8.1: Nombramiento de vainilla individual para cada caja.

8.6. LISTA DE FIGURAS

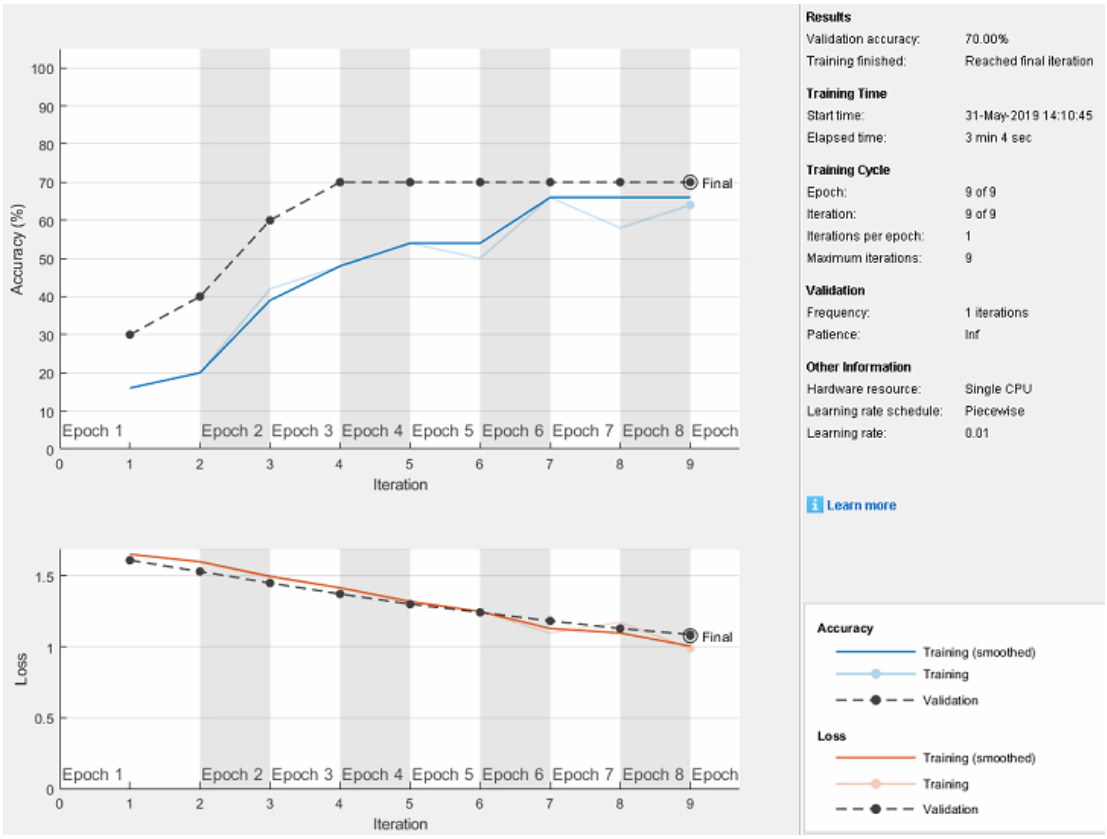

Figura 8.1: Exactitud y costo computacional para 10 imágenes.

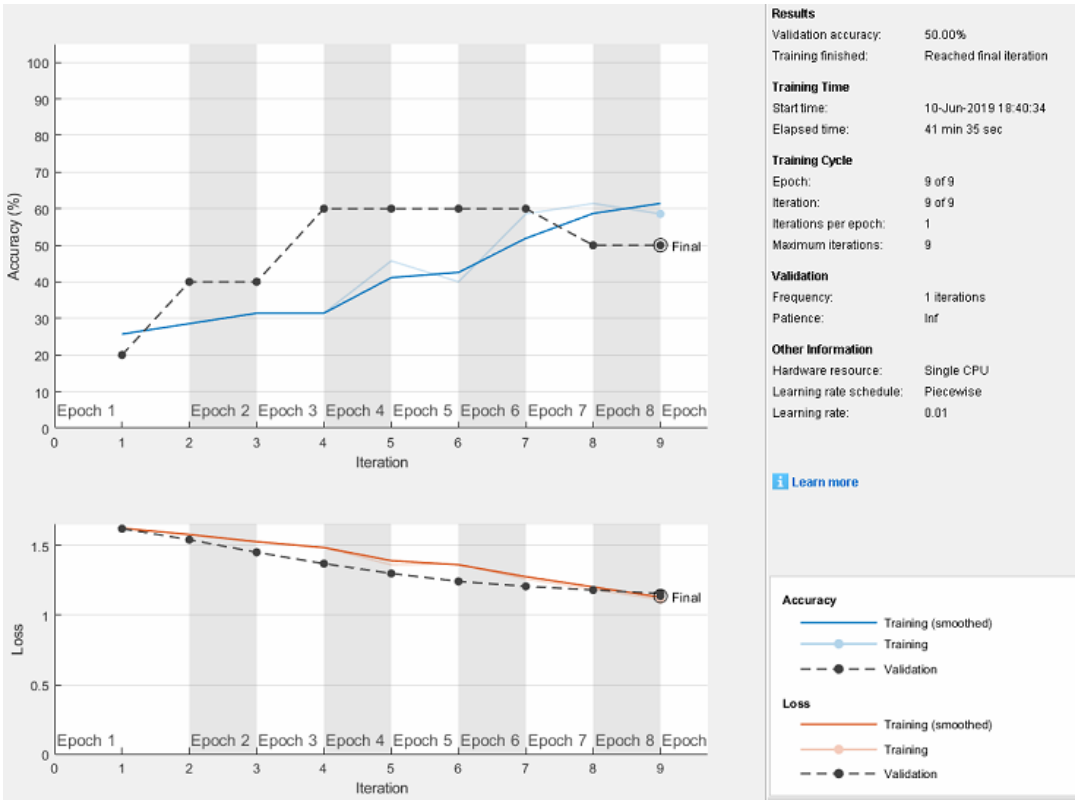

Figura 8.2: Exactitud y costo computacional para 15 imágenes.

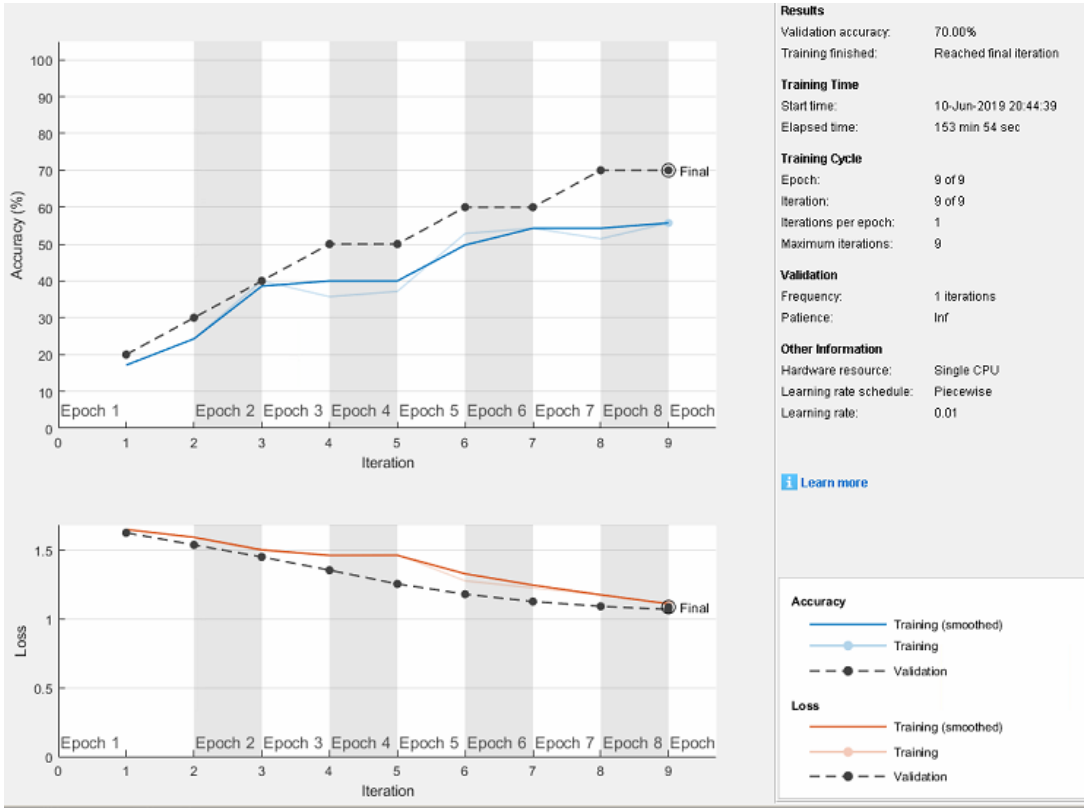

Figura 8.3: Exactitud y costo computacional para 20 imágenes.

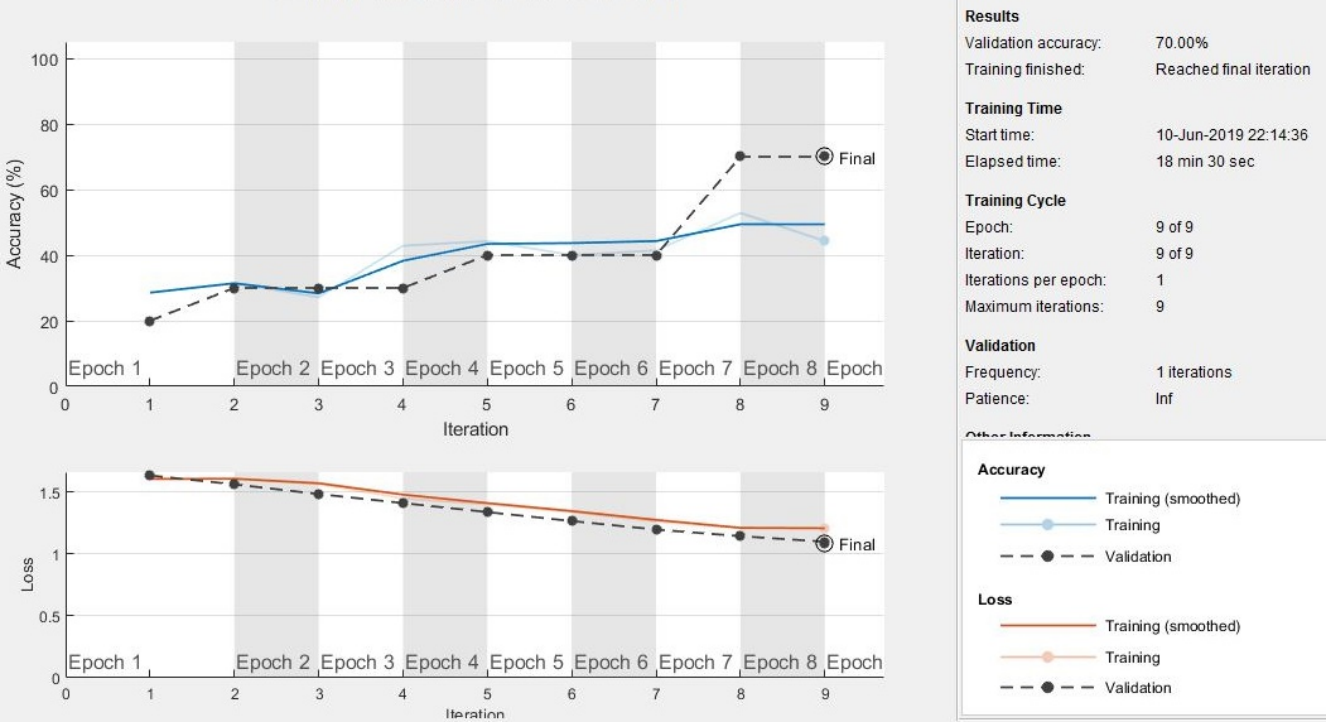

Figura 8.4: Exactitud y costo computacional para 25 imágenes.

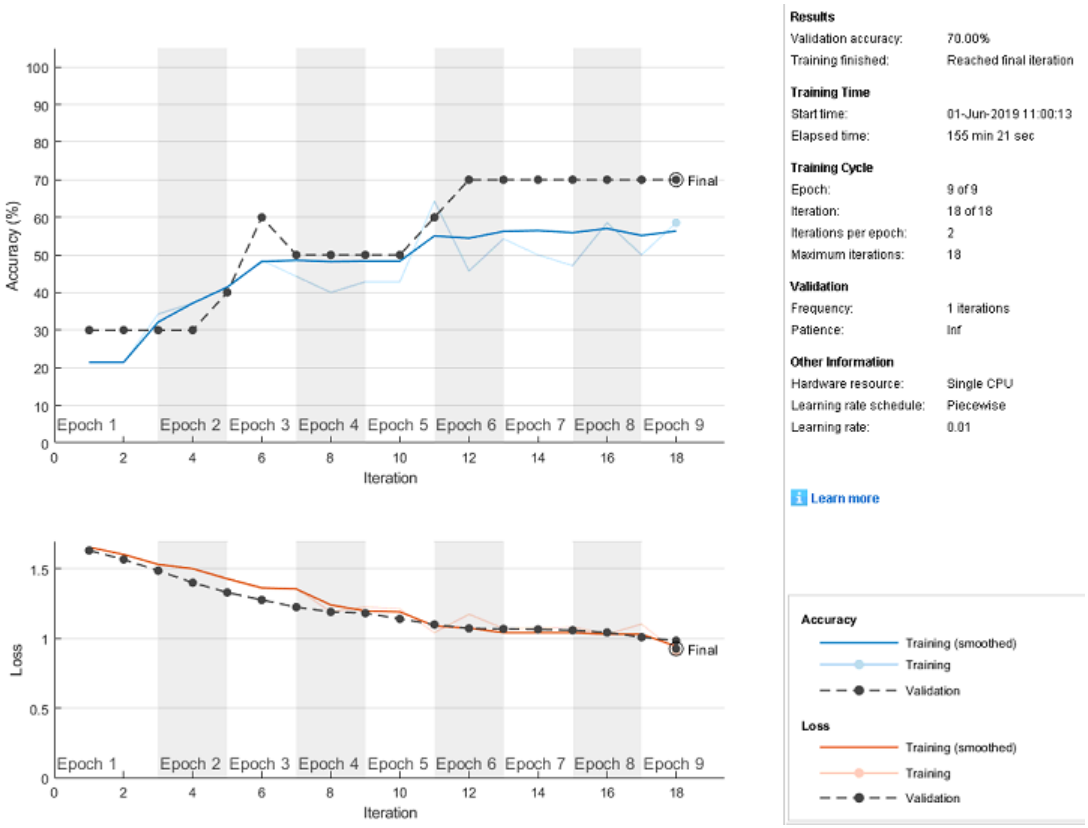

Figura 8.5: Exactitud y costo computacional para 30 imágenes.

8.6.1. Toma de imágenes

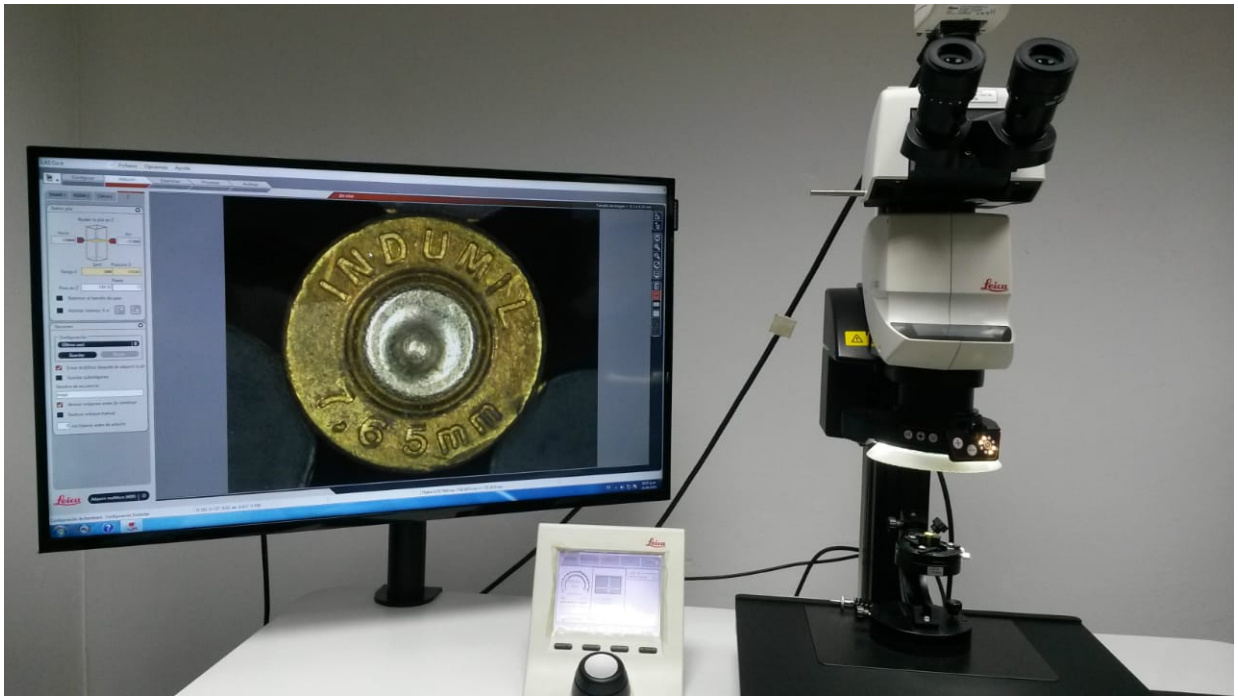

Figura 8.6: Estereoscopio M205A e interfaz del programa Leica Application Suite.

Figura 8.7: Lámpara del estereoscopio iluminada por segmentos.

8.6.2. Matrices de confusión SURF

Predicciones	Arma 1	258	292	233	187	203
	Arma 2	229	277	195	193	213
	Arma 3	200	236	208	210	193
	Arma 4	209	204	283	297	292
	Arma 5	252	221	311	343	329
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.8: Clasificador KNN Euclidiana.

Predicciones	Arma 1	247	306	209	204	212
	Arma 2	259	270	194	190	239
	Arma 3	215	255	215	185	212
	Arma 4	205	194	319	287	261
	Arma 5	222	205	193	364	306
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.9: Clasificador KNN City Block.

Predicciones	Arma 1	473	486	456	431	448
	Arma 2	280	326	285	295	291
	Arma 3	249	242	266	239	263
	Arma 4	81	85	127	134	110
	Arma 5	65	91	96	131	118
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.10: Clasificador KNN Mahalanobis.

Predicciones	Arma 1	131	103	56	36	76
	Arma 2	84	78	40	22	60
	Arma 3	92	93	80	66	61
	Arma 4	24	9	23	22	20
	Arma 5	817	947	1031	1084	1013
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.11: Clasificador SVM Fine.

Predicciones	Arma 1	413	331	160	144	184
	Arma 2	298	429	108	122	154
	Arma 3	202	228	407	149	186
	Arma 4	120	120	294	479	253
	Arma 5	115	122	251	336	453
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.12: Clasificador SVM Medium.

8.6.3. Matrices de confusión CNN

Predicciones	Arma 1	0,5	0,0	0,5	0,0	0,0
	Arma 2	0,0	0,5	0,0	0,0	0,0
	Arma 3	0,5	0,5	0,0	0,0	0,0
	Arma 4	0,0	0,0	0,0	1,0	0,5
	Arma 5	0,0	0,0	0,5	0,0	0,5
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.13: Validación de prueba para 15 imágenes.

Predicciones	Arma 1	1,0	0,5	0,5	0,0	0,0
	Arma 2	0,0	0,5	0,0	0,0	0,0
	Arma 3	0,0	0,0	0,0	0,0	0,0
	Arma 4	0,0	0,0	0,0	1,0	0,0
	Arma 5	0,0	0,0	0,5	0,0	1,0
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.14: Validación de prueba para 20 imágenes.

Predicciones	Arma 1	1,0	0,0	0,5	0,0	0,0
	Arma 2	0,0	1,0	0,0	0,0	0,0
	Arma 3	0,0	0,0	0,0	0,0	0,0
	Arma 4	0,0	0,0	0,5	1,0	0,5
	Arma 5	0,0	0,0	0,0	0,0	0,5
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.15: Validación de prueba para 25 imágenes.

Predicciones	Arma 1	1,0	0,0	0,0	0,0	0,0
	Arma 2	0,0	0,5	0,0	0,0	0,0
	Arma 3	0,0	0,5	0,5	0,0	0,5
	Arma 4	0,0	0,0	0,0	1,0	0,0
	Arma 5	0,0	0,0	0,5	0,0	0,5
		Arma 1	Arma 2	Arma 3	Arma 4	Arma 5
		Reales				

Figura 8.16: Validación de prueba para 30 imágenes.

8.6.4. Clasificación técnica alternativa (Análisis morfológico)

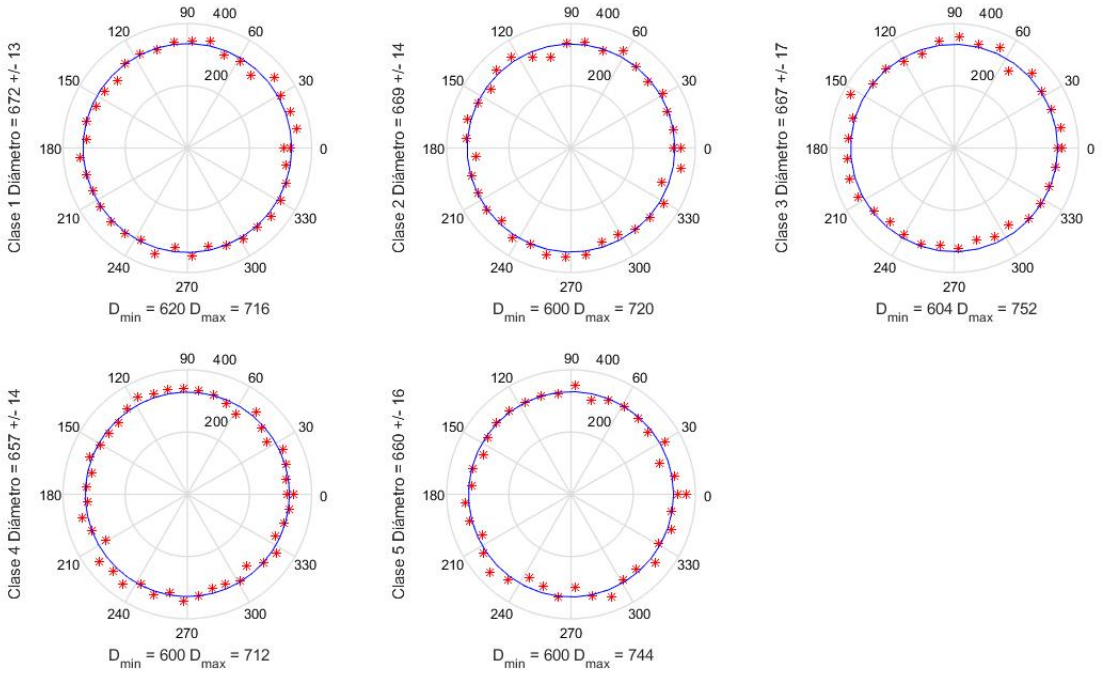

Figura 8.17: Ubicación del diámetro del cráter de percusión alrededor del promedio.

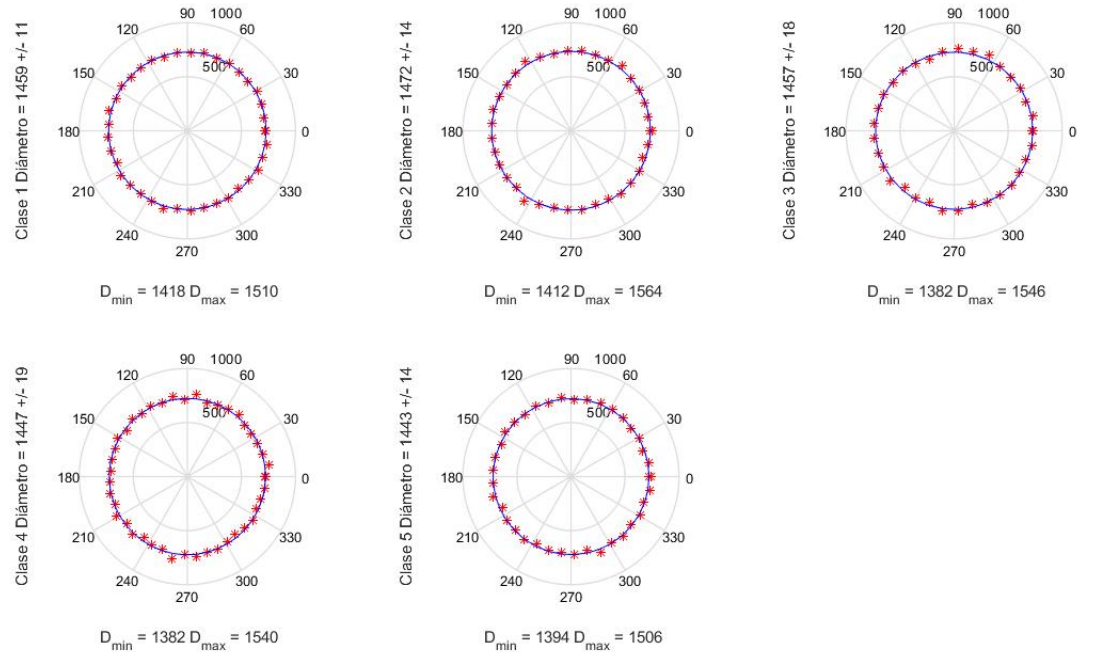

Figura 8.18: Ubicación del diámetro del fulminante alrededor del promedio.

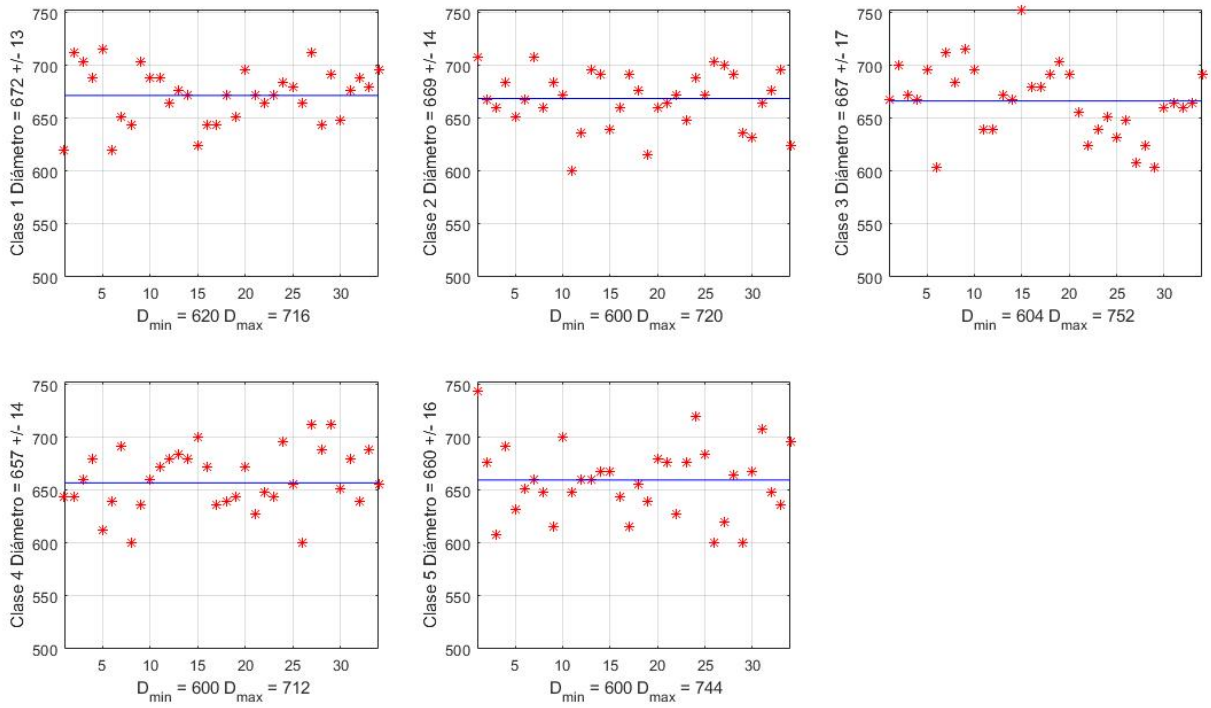

Figura 8.19: Ubicación en línea del diámetro del cráter de percusión alrededor del promedio.

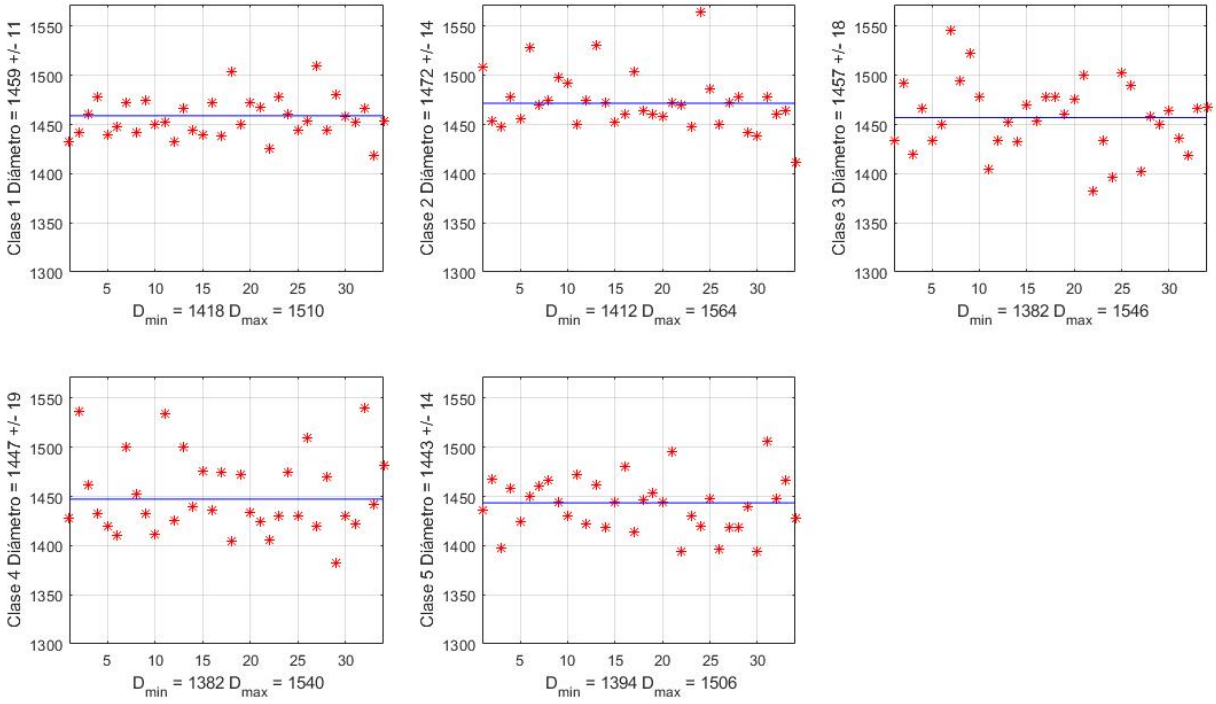

Figura 8.20: Ubicación en línea del diámetro del fulminante alrededor del promedio.

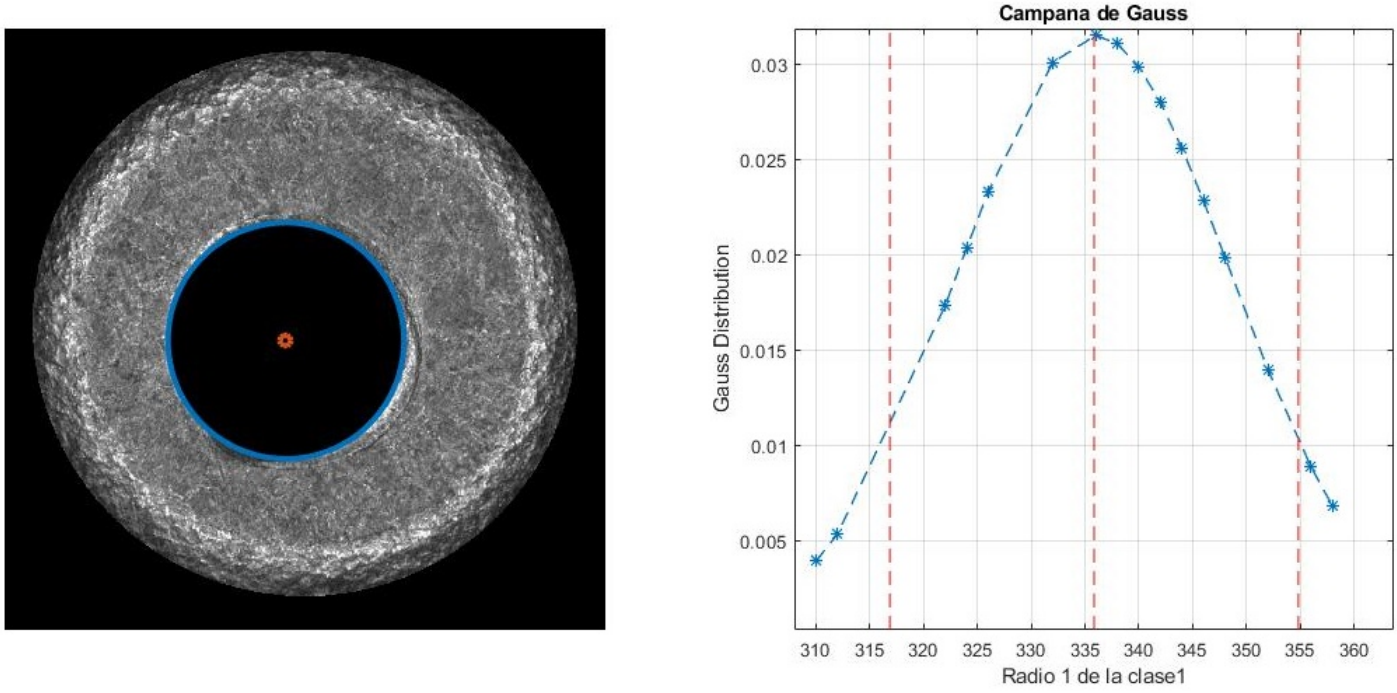

Figura 8.21: Campana de Gauss para radio del cráter de percusión en la clase 1.

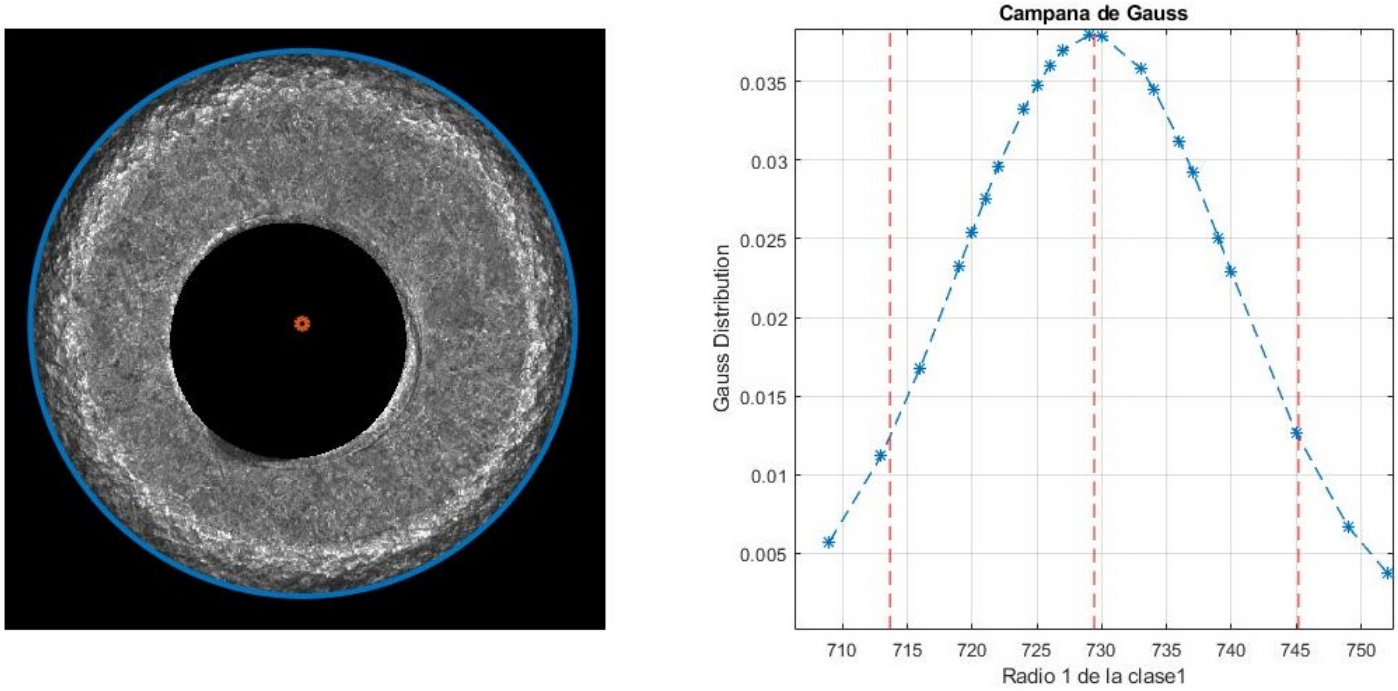

Figura 8.22: Campana de Gauss para radio del fulminante en la clase 1.

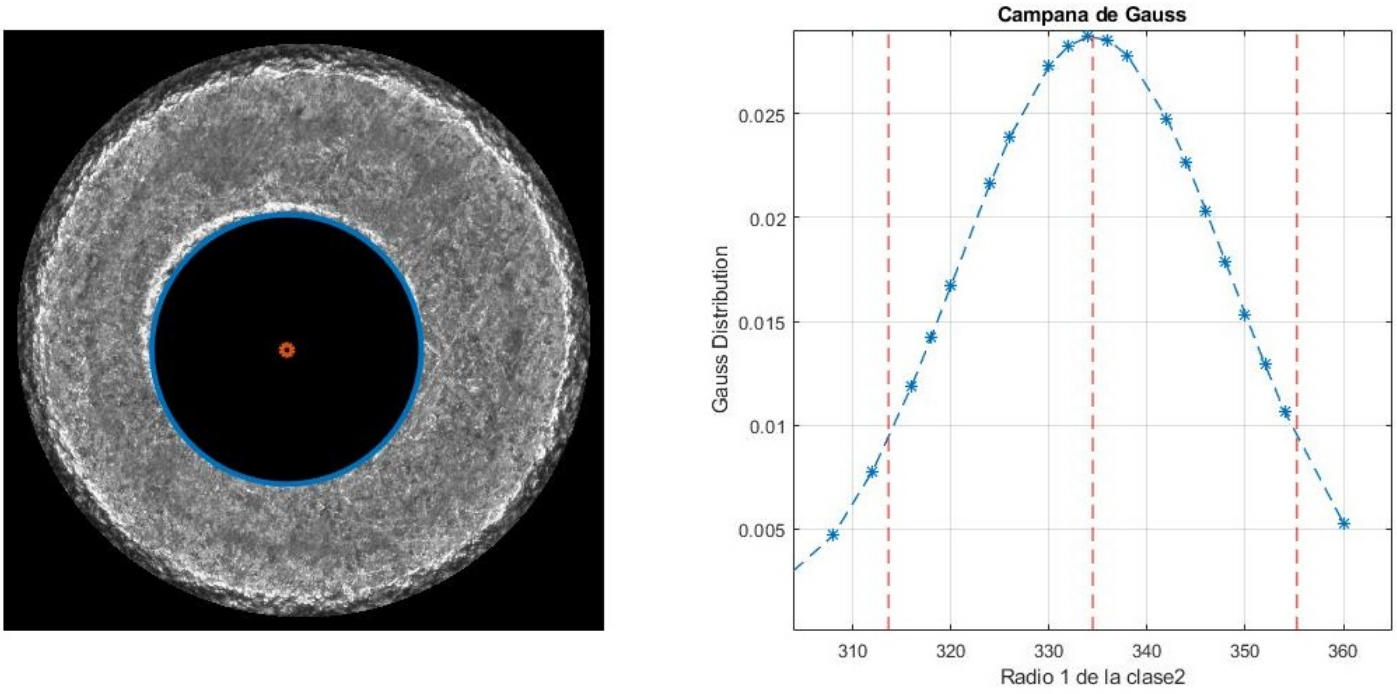

Figura 8.23: Campana de Gauss para radio del cráter de percusión en la clase 2.

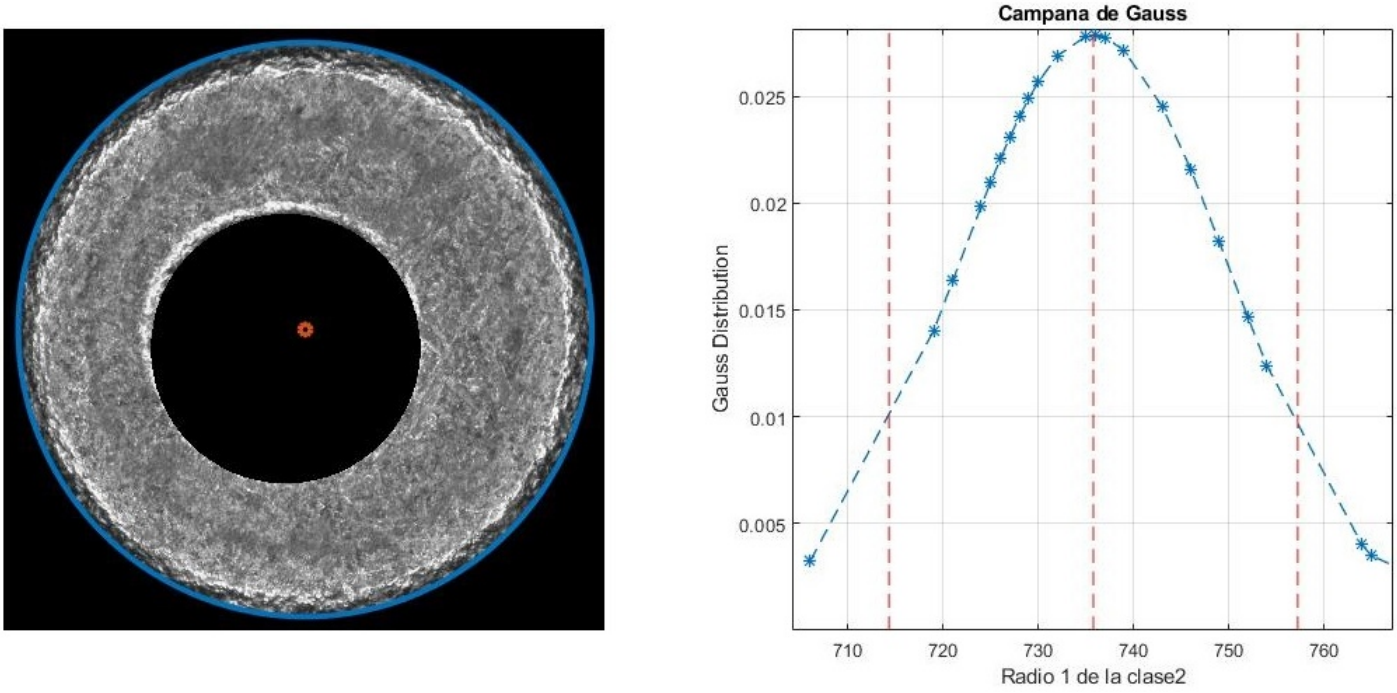

Figura 8.24: Campana de Gauss para radio del fulminante en la clase 2.

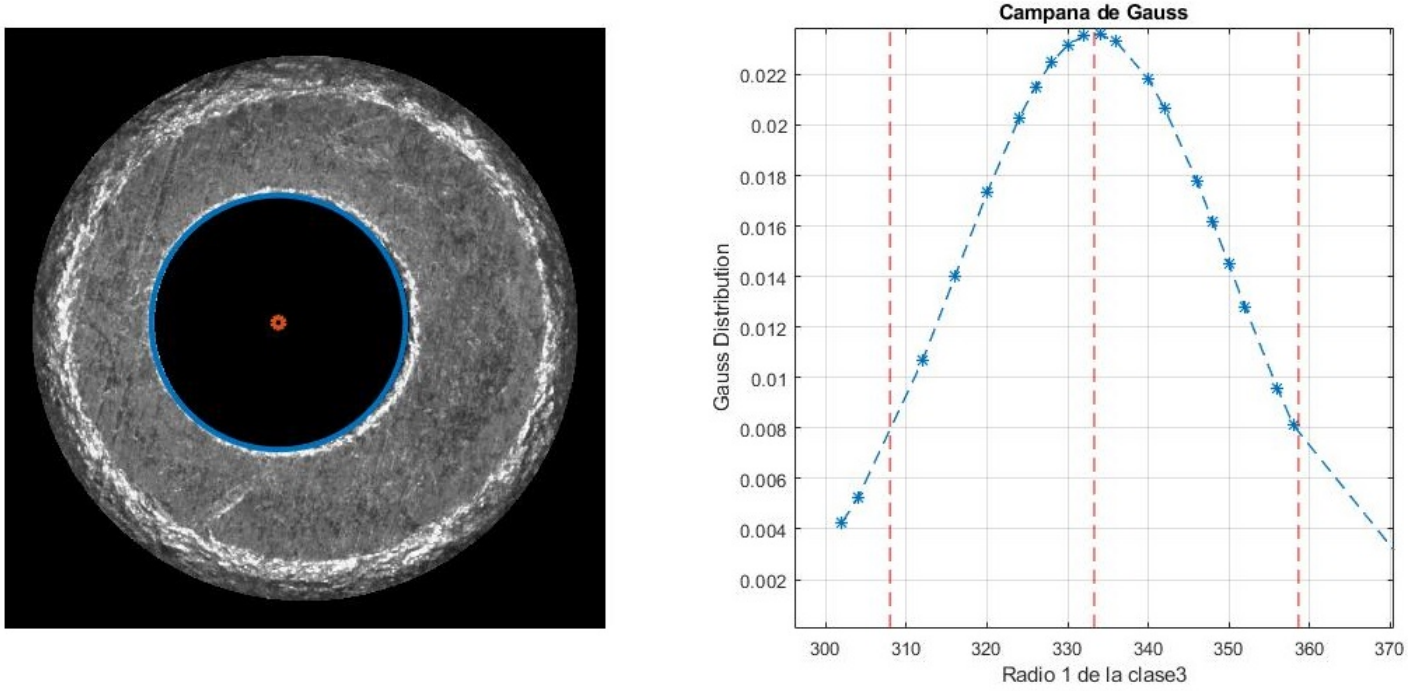

Figura 8.25: Campana de Gauss para radio del cráter de percusión en la clase 3.

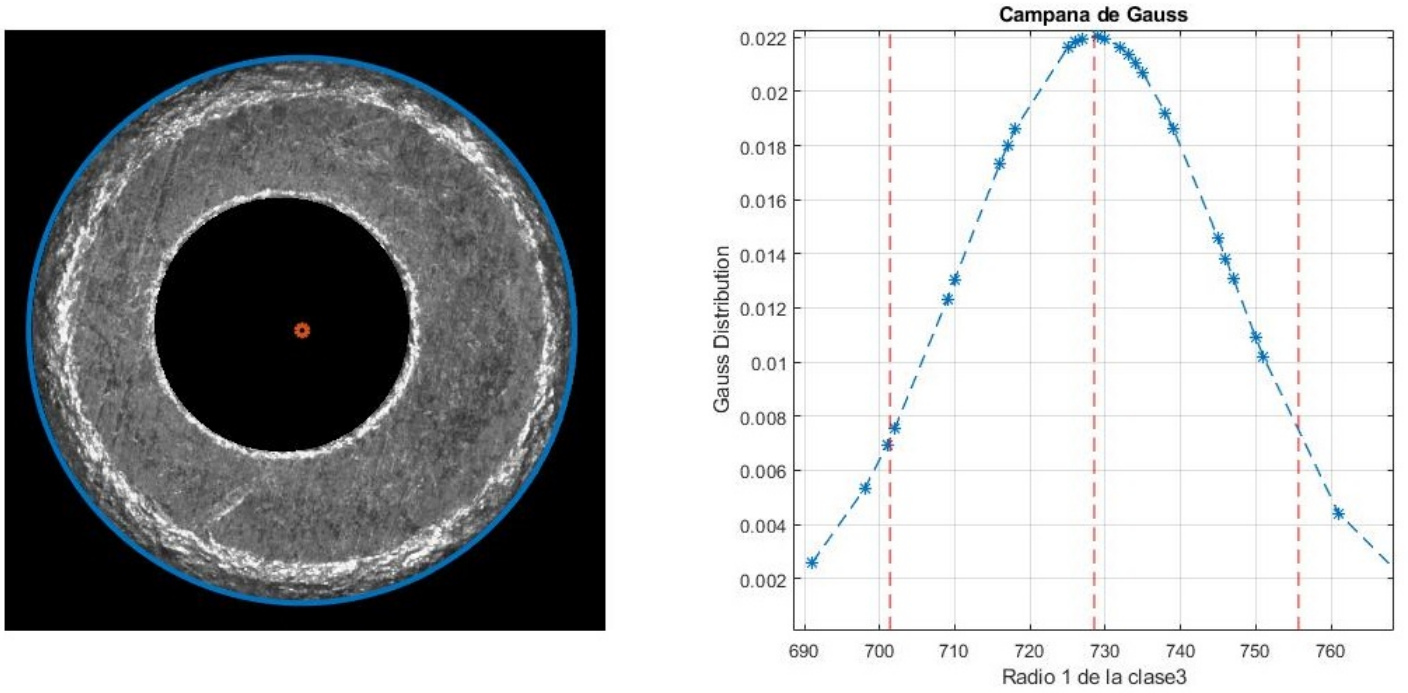

Figura 8.26: Campana de Gauss para radio del fulminante en la clase 3.

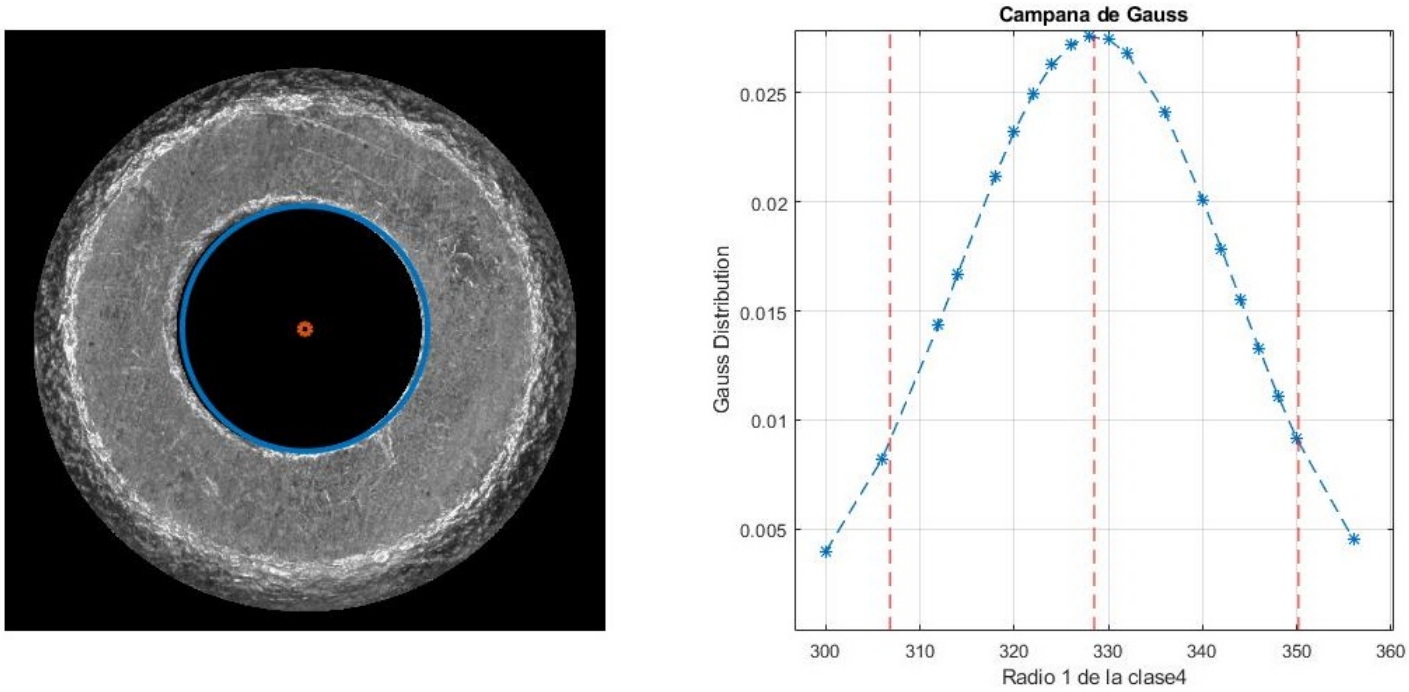

Figura 8.27: Campana de Gauss para radio del cráter de percusión en la clase 4.

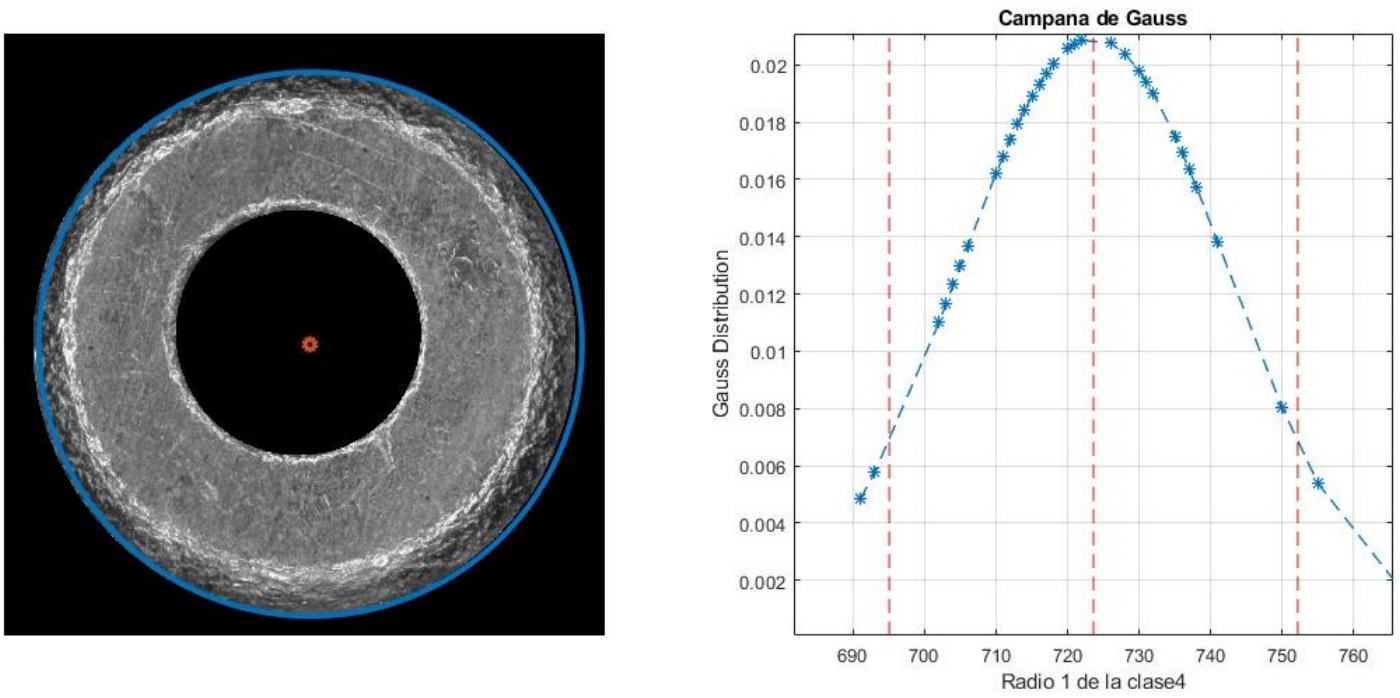

Figura 8.28: Campana de Gauss para radio del fulminante en la clase 4.

Bibliografía

[1] K. Aguirre and J. A. Restrepo, "El control de armas como estrategia de reducción de la violencia en colombia: pertinencia, estado y desafíos," *Revista Criminalidad*, vol. 52, no. 1, pp. 265–284, 2010.

[2] J. Méndez García, J. H. Rivera Piedrahita, and J. A. Soto Mejía, "Reconocimiento de texturas en imágenes de proyectiles: Un aporte a la identificación automática de armas," *Ciencia e Ingeniería Neogranadina*, vol. 22, no. 1, 2012.

[3] Z. J. Geradts, J. Bijhold, and R. Hermsen, "Pattern recognition in a database of cartridge cases," in *Investigation and Forensic Science Technologies*, vol. 3576. International Society for Optics and Photonics, 1999, pp. 104–116.

[4] N. A. M. Ghani, C.-Y. Liong, and A. A. Jemain, "Analysis of geometric moments as features for identification of forensic ballistics specimen," in *International Work-Conference on Artificial Neural Networks*. Springer, 2009, pp. 604–611.

[5] S. B. A. Kamaruddin, N. A. M. Ghani, C.-Y. Liong, and A. A. Jemain, "Firearm recognition based on whole firing pin impression image via backpropagation neural network," in *2011 International Conference on Pattern Analysis and Intelligence Robotics*, vol. 1. IEEE, 2011, pp. 177–182.

[6] N. A. M. Ghani, S. B. A. Kamaruddin, C.-Y. Liong, and A. A. Jemain, "Firearm identification using numerical features of centre firing pin impression image," in *2012 International Symposium on Computer Applications and Industrial Electronics (ISCAIE)*. IEEE, 2012, pp. 293–296.

[7] I. Fogg and N. Paul, "Forensics image background matching using scale invariant feature transform (sift) and speeded up robust features (surf)," AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH DEPT OF ELECTRICAL AND âŠ, Tech. Rep., 2007.

[8] J. M. Beattie, *The first English detectives: the Bow Street Runners and the policing of London, 1750-1840*. Oxford University Press, 2012.

[9] W. G. Eckert, "Historical development of forensic sciences," *Introduction to Forensic Science*, p. 20, 1997.

[10] C. H. Goddard, "Scientific identification of firearms and bullets," *Am. Inst. Crim. L. & Criminology*, vol. 17, p. 254, 1926.

[11] W. G. Eckert, "Historical development of forensic sciences," *Introduction to Forensic Science*, p. 20, 1997.

[12] A. A. Kling, *Ballistics*. Greenhaven Publishing LLC, 2008.

[13] W. J. Kaiser and L. D. Bell, "Tunnel and field effect carrier ballistics," Apr. 18 1989, uS Patent 4,823,004.

[14] T. Warlow, *Firearms, the law, and forensic ballistics*. CRC Press, 2004.

[15] N. A. M. Ghani, C.-Y. Liong, and A. A. Jemain, "Neurocomputing approach for firearm identification." *Pertanika Journal of Science & Technology*, vol. 26, no. 1, 2018.

[16] I. H. García, *Estética, ciencia y tecnología: creaciones electrónicas y numéricas*. Pontificia Universidad Javeriana, 2005.

[17] R. C. Gonzalez, R. E. Woods *et al.*, "Digital image processing [m]," *Publishing house of electronics industry*, vol. 141, no. 7, 2002.

[18] P. Panchal, S. Panchal, and S. Shah, "A comparison of sift and surf," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 323–327, 2013.

[19] B. J. Copeland and D. Proudfoot, "Alan turingâs forgotten ideas in computer science," *Scientific American*, vol. 280, no. 4, pp. 98–103, 1999.

[20] J. D. Cowan, "Neural networks: the early days," in *Advances in neural information processing systems*, 1990, pp. 828–842.

[21] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[22] D. J. Matich, "Redes neuronales: Conceptos básicos y aplicaciones," *Universidad Tecnológica Nacional, México*, 2001.

[23] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[24] G. Li and J. Shi, "On comparing three artificial neural networks for wind speed forecasting," *Applied Energy*, vol. 87, no. 7, pp. 2313–2320, 2010.

[25] R. M. Golden, "The âbrain-state-in-a-boxâ neural model is a gradient descent algorithm," *Journal of Mathematical Psychology*, vol. 30, no. 1, pp. 73–80, 1986.

[26] D. Li, "A new approach for firearm identification with hierarchical neural networks based on cartridge case images," in *2006 5th IEEE International Conference on Cognitive Informatics*, vol. 2. IEEE, 2006, pp. 923–928.

[27] R. J. Locles, *Tratado de balística*. La Rocca, 2003.

[28] Á. Burgos, "La criminalística y su importancia en el campo forense," *Revista Digital de la Maestría en Ciencias Penales*, no. 2, pp. 239–239, 2010.

[29] M. Denny, "The internal ballistics of an air gun," *The Physics Teacher*, vol. 49, no. 2, pp. 81–83, 2011.

[30] D. A. Volgas, J. P. Stannard, and J. E. Alonso, "Ballistics: a primer for the surgeon," *Injury*, vol. 36, no. 3, pp. 373–379, 2005.

[31] J. R. Manzano-Trovamala Figueroa, M. G. G. Molina, and F. A. Velazco, "Balística: Balística de efectos o balística de las heridas," *Cirujano General*, vol. 23, no. 4, pp. 266–272, 2001.

[32] R. G. Nichols, "Firearm and toolmark identification criteria: A review of the literature," *Journal of Forensic Science*, vol. 42, no. 3, pp. 466–474, 1997.

[33] M. Bonfanti *et al.*, "The influence of manufacturing processes on the identification of bullets and cartridge cases–a review of the literature." *Science & justice: journal of the Forensic Science Society*, vol. 39, no. 1, pp. 3–10, 1999.

[34] J. M. García, J. H. Rivera, and J. S. Mejía, "Extracción de características de textura para cotejo de proyectiles en balística." *Scientia et technica*, vol. 1, no. 44, pp. 229–233, 2010.

[35] C. Jiménez and J. Javier, "La balística forense como herramienta fundamental para la identificación del arma de fuego utilizada en un hecho punible," 2015.

[36] B. Jähne, *Digital image processing: concepts, algorithms, and scientific applications*. Springer Heidelberg, 1997, vol. 6.

[37] J.-M. Morel and S. Solimini, *Variational methods in image segmentation: with seven image processing experiments*. Springer Science & Business Media, 2012, vol. 14.

[38] D. Kim and R. Dahyot, "Face components detection using surf descriptors and svms," in *2008 International Machine Vision and Image Processing Conference*. IEEE, 2008, pp. 51–56.

[39] V. M. Anees, G. S. Kumar, and M. Sreeraj, "Automatic image annotation using surf descriptors," in *2012 Annual IEEE India Conference (INDICON)*. IEEE, 2012, pp. 920–924.

[40] P. F. Alcantarilla, L. M. Bergasa, and A. J. Davison, "Gauge-surf descriptors," *Image and Vision Computing*, vol. 31, no. 1, pp. 103–116, 2013.

[41] D. J. Matich, "Redes neuronales: Conceptos básicos y aplicaciones," *Universidad Tecnológica Nacional, México*, 2001.

[42] N. Sahla, "A deep learning prediction model for object classification," 2018.

[43] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[45] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.

[46] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[47] A. Hay, "The derivation of global estimates from a confusion matrix," *International Journal of Remote Sensing*, vol. 9, no. 8, pp. 1395–1398, 1988.

[48] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[49] D. Scherer, H. Schulz, and S. Behnke, "Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors," in *International conference on Artificial neural networks*. Springer, 2010, pp. 82–91.

[50] C. Platero, "Apuntes de visión artificial," *Departamento de Electrónica, Automática e Informática Industrial*, 2009.

[51] M. Herrera and A. Moreno, "Análisis morfológico en el procesamiento óptico-digital de imágenes para el diagnóstico de la disquinesia ciliar," *Información tecnológica*, vol. 25, no. 1, pp. 33–40, 2014.

[52] Y. Morales Olivera, J. García Parrado, P. E. Reyes Fernández, and J. V. Lorenzo Ginori, "Experiencias en la implementación de las operaciones morfológicas de erosión y dilatación para imágenes binarias empleando vecindades adaptativas," *Ingeniería Electrónica, Automática y Comunicaciones*, vol. 33, no. 2, pp. 34–41, 2012.